Architecture du plan de contrôle SDN et placement de services réseaux dans les infrastructures des opérateurs

Jean-Michel Sanner

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

THESE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Télécommunications*

Par

« Jean-Michel SANNER »

« Architecture du plan de contrôle SDN et placement des services réseaux dans les infrastructures des opérateurs»

Thèse présentée et soutenue à « RENNES », le « 23 Juillet 2019 »
Unité de recherche : Dyonisos
Thèse N° :

Rapporteurs avant soutenance :

Hossam Afifi, Professeur, Institut Mines Télécom/Télécom Sud Paris,
Thierry Turletti, Directeur de recherche, INRIA Sophia Antipolis – Méditerranée, Sophia Antipolis

Composition du Jury :

Président :	Prénom Nom	Fonction et établissement d'exercice (9)*(à préciser après la soutenance)*
Rapporteurs :	Hossam Afifi	Professeur, Institut Mines Télécom Sud Paris
	Thierry Turletti	Directeur de recherche, INRIA Sophia- Antipolis
Examinateurs :	Rami Langar,	Professeur, Université Paris-Est, Marne-la-Vallée
	Meryem Ouzzif,	Ingénieure de recherche, Orange-Labs, Rennes
	Nancy Perrot,	Ingénieure de recherche, Orange-Labs, Paris
Dir. de thèse :	Gerardo Rubino,	Directeur de recherche, INRIA, Rennes
Co-dir. de thèse :	Yassine Hadjadj-Aoul,	Maître de conférence, Université de RENNES I, Rennes

An dianav a rog ac'hanon (l'inconnu me dévore)

Remerciements

Il n'est pas très facile de s'engager dans un travail de thèse si tardivement dans sa vie professionnelle. Heureusement des gens étaient là pour m'aider. Je tiens donc à adresser mes remerciements à:

tous les membres des deux équipes à Orange et B<>COM et leurs responsables, dans lesquelles j'ai travaillé pendant le déroulement de cette thèse et qui m'ont accompagné et supporté.

Meryem Ouzzif et Yassine Hadjadj-Aoul pour leur soutien constant et attentif tout au long de ce travail et qui sont je pense devenus un peu des amis.

Gerardo Rubino pour ses explications d'éminent mathématicien.

Quang Pham Tran Anh pour la qualité de sa collaboration et sa grande disponibilité dans le cadre du postdoc qu'il a effectué à BCOM et aussi à Pierrick Louin pour à peu près les mêmes raisons dans le cadre d'Orange-Labs.

Pierre Collet pour son cours remarquable sur les algorithmes évolutionnaires sur la plateforme FUN.

Et enfin et surtout à Katelin qui m'a accompagné au quotidien tout au long de cette histoire pourtant très personnelle...

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction générale

“ 'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves, And
the mome raths outgrabe. ”

Lewis Caroll

Sommaire

1.1 Introduction

En 2008, [McKeown et collab., 2008] publiaient le papier intitulé: "Enabling Innovation in Campus Networks". Les auteurs proposaient le protocole OpenFlow (OF) comme une solution permettant ainsi aux constructeurs d'offrir une interface de programmation à leurs équipements réseaux permettant aux chercheurs d'expérimenter de nouvelles approches et de nouveaux protocoles réseaux.

OF proposait une solution pour séparer plan de données et plan de contrôle qui soit facile à mettre à mettre en œuvre par les industriels dans leurs équipements. Cette séparation permettait aussi de centraliser le plan de contrôle en introduisant dans le réseau une fonction de contrôleur centralisée, chargée de piloter le plan de données.

Cette proposition déclencha une dynamique profonde de remise en question des architectures réseaux traditionnelles. Cependant, cette approche n'était pas si nouvelle. Des propositions similaires avaient fait par exemple l'objet de travaux de standardisation à l'IETF.

C'était déjà le cas en 2002 avec la RFC 5810 ForCes ou [Halpern et collab., 2010] distinguaient notamment dans les équipements les "Forwarding Elements" (FE) et les "Control Elements" (CE). L'approche initiale intitulée ForCes prévoyait aussi la centralisation du plan de contrôle mais portait surtout l'idée de séparer plan de contrôle et plan de données, en favorisant ainsi l'indépendance entre les deux plans et en stimulant ainsi la concurrence et les propositions des industriels.

L'introduction d'un contrôle centralisé associé à une certaine intelligence était aussi présente dans le concept de Policy Based Networking (PBN) introduit à l'IETF via la notion de Policy Decision Point (PDP) et de Policy Enforcement Point (PEP) qui est mise en oeuvre pour provisionner et appliquer des politiques de QoS. Ces approches et d'autres encore plus récentes ont été ensuite désignées et réunies sous le sigle SDN.

Tandis que la recherche cherchait à préciser les limites et les potentialités de l'approche SDN OF dans le monde des réseaux physiques ou virtuels et dans les datacenters, en 2012 est arrivé le paradigme ETSI-NFV qui s'appuyait lui sur les potentialités de la virtualisation pour proposer de faire évoluer les infrastructures des réseaux des opérateurs de télécommunication. L'idée était en particulier de faire supporter les fonctions et services réseaux par un environnement virtuel dans des serveurs banalisés.

Les opérateurs de télécommunication se sont emparés de ces deux approches pour tenter de faire évoluer leurs infrastructures avec deux objectifs, réduire les coûts d'exploitation et améliorer la flexiblité et la programmabilité de l'infrastructure.

1.2 Motivations

Cette évolution nécessite cependant de lever de nombreux verrous technologiques.

Pour ce qui concerne le SDN, il s'agit d'une évolution pratiquement inverse de la tendance à la distribution des fonctions qui avait prévalu jusqu'à aujourd'hui dans les infrastructures réseaux et qui a permis de disposer de réseaux fiables et résilients. L'enjeu devient donc paradoxalement de savoir comment ne pas perdre avec une architecture SDN centralisée les avantages d'un plan de contrôle distribué.

Pour la virtualisation, il est possible d'identifier au moins trois verrous significatifs liés à la mise en oeuvre d'équipements banalisés.

Ce sont la consommation énergétique dont il faut prouver qu'elle n'augmentera pas du fait de la virtualisation, c'est aussi la fiabilité qui doit restée élevée malgré la mise en oeuvre de logiciels potentiellement moins fiables sur des équipements informatiques

standards mais éventuellement inadaptés et c'est enfin les performances réseaux que la virtualisation pénalise inévitablement.

Les potentialités de ces technologies font donc aussi naître de nouvelles questions concernant la mise à l'échelle, la fiabilité, la résilience et les performances que l'on pourra obtenir pour les services réseaux.

Dans ce contexte deux problématiques sont abordées dans la thèse.

La première problématique traite d'une part de l'approche centralisée et des limitations du protocole OF et d'autre part des contraintes de la virtualisation des fonctions réseaux pour lesquelles il n'est pas évident de réaliser la même richesse fonctionnelle de services ni d'obtenir les mêmes performances que celles des réseaux classiques.

Dans ces conditions, cela pose la question de la pertinence des différentes promesses portées par SDN et Network Function Virtualisation (NFV) : élasticité, scalabilité, évolutivité.

Pour répondre à cette première problématique, la thèse propose une architecture SDN distribuée, permettant la composition, puis la validation et le déploiement de services réseaux reconfigurables dynamiquement et de manière différenciée et prenant en compte les Service Level Agrement (SLA) associés aux services.

La démarche s'appuie sur un plan de contrôle et de management adaptés aux fonctionnalités enrichies par rapport à OF, permettant de programmer un nœud réseau générique programmable banalisé. Cela est réalisé à partir d'une fonction de contrôle organisée de manière hiérarchique avec des parties locales et une partie centralisée.

Nous illustrons ensuite une partie des idées proposées dans une preuve de concept.

La seconde problématique part du constat tiré de la première qu'il faut réaliser un placement différencié des fonctions réseaux virtualisées de contrôle ou de traitement des paquets dans l'infrastructure IT qui porte les services.

C'est en particulier grâce à l'optimisation de ce placement que l'on pourra respecter les SLAs des services réseaux déployés et optimiser les coûts de gestion et d'exploitation de l'infrastructure.

Les algorithmes de placement des fonctions réseaux dans les infrastructures SDN et NFV doivent permettre de déployer les services, à la demande et au rythme nécessaire. Ils doivent aussi être capables de répondre à des besoins évolutifs et variables dans le temps. Il peut s'agir par exemple d'une évolution des exigences en terme de SLA ou bien d'une évolution du contexte d'exploitation, comme par exemple la charge du réseau, ou encore d'une évolution de configuration ou enfin de pannes qui interviennent dans le réseau.

Pour répondre à cette seconde problématique, la thèse explore différentes approches algorithmiques envisageables afin de réaliser le placement des fonctions réseaux de manière efficace, rapide et flexible. Nous nous intéressons en particulier aux métaheuristiques stochastiques évolutionnaires.

L'objectif final à partir de ces travaux vise à proposer un framework algorithmique de placement générique qui pourrait s'adapter potentiellement à différents contextes d'allocations de ressources, du placement de contrôleurs ou de réseaux virtuels au placement de chaînes de fonctions réseaux virtualisées.

1.3 Contributions de la thèse

Les différentes contributions de la thèse sont répertoriées dans le tableau 1.3 par ordre chronologique. Deux contributions portent sur l'architecture des réseaux SDN. Les autres contributions sont centrées sur les algorithmes et la problématique du placement de fonctions réseaux.

Architecture SDN

La première contribution décrit une architecture SDN, distribuée et dynamique orientée services réseaux qui s'efforce de faire la synthèse de différents concepts visant à lever les verrous techniques identifiés propres aux architecture SDN. Elle a fait l'objet d'un article présenté à la conférence NetSoft 2015.

Cette contribution a été complétée par une preuve de concept illustrant une partie des points clés de l'architecture proposée. Plus précisément, elle montre le déploiement de modèles de fonctions réseaux ditribuée, de manière dynamique dans un architecture virtualisée et la gestion de leur cycle de vie. Cette preuve de concept a fait l'objet d'un article présenté à la conférence Saconet 2018.

Placement de contrôleurs

Une première contribution dans le volet "placement de contrôleurs" consiste en une heuristique permettant de réaliser un placement optimisé de contrôleurs de manière très rapide même dans des réseaux de taille importante, de l'ordre de quelques milliers de nœuds. Cette contribution a fait l'objet d'un article présenté à la conférence GIIS 2016.

Ensuite, l'analyse de la faisabilité de la résolution d'un problème de placement de contrôleurs avec un algorithme évolutionnaire a été présentée sous forme d'un poster à la conférence SDN days 2016.

Nous avons également effectué à SDN Days 2017 une présentation orale visant à mettre en valeur l'intérêt des algorithmes évolutionnaires comme outil générique de résolution de problèmes de placement de services réseaux.

Enfin, nous avons présenté à la conférence CNSM 2017, une proposition d'un algorithme évolutionnaire multi-objectifs permettant de placer des contrôleurs SDN dans un réseau en intégrant un objectif visant à optimiser la fiabilité du réseau SDN.

Placement de service réseaux

Nous nous sommes ensuite intéressés à l'expérimentation de l'utilisation des algorithmes évolutionnaires pour le placement de services réseaux. Cela a été réalisé en collaboration avec M. Quang Phan Tran Anh, qui a effectué un postdoc à B-COM.

Cette collaboration a permis de produire un premier papier que nous avons présenté dans un keynote comme papier invité à la conférence Saconet 2018. Les travaux présentés dans ce papier visent à montrer les performances des algorithmes évolutionnaires pour résoudre le problème du routage multi-objectifs et multi-contraintes.

Les potentialités de ce type d'algorithmes étant démontrées, nous avons étendu la démarche au problème de placement de services réseaux définis par des Virtual Network Function Forwarding Graph (VNF-FG)s. Les résultats de ces travaux doivent faire l'objet d'une publication dans le journal Wiley IJCS.

Présentation et publications dans l'ordre chronologique

Travaux et publications réalisés
DICES: A dynamic adaptive service-driven SDN architecture - Jean-Michel Sanner ; Meryem Ouzzif ; Yassine Hadjadj-Aoul - Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft) - London, UK, 2015
Evolutionary algorithms for optimized SDN controllers & NVFs' placement in SDN networks - Jean-Michel Sanner ; Meryem Ouzzif ; Yassine Hadjadj-Aoul - Poster in SDN days, Paris, France, 2016
Hierarchical clustering for an efficient controllers' placement in software defined networks - Jean-Michel Sanner ; Yassine Hadjadj-Aoufi ; Meryem Ouzzif ; Gerardo Rubino - Global Information Infrastructure and Networking Symposium (GIIS) - Porto, Portugal, 2016
Multi-objectives genetics algorithms for placement problem - Jean-Michel Sanner ; Meryem Ouzzif ; Yassine Hadjadj-Aoul - Presentation, SDN days, Paris, France, 2017
An evolutionary controllers' placement algorithm for reliable SDN networks - Jean-Michel Sanner ; Yassine Hadjadj-Aoul ; Meryem Ouzzif ; Gerardo Rubino - International Workshop on Management of SDN and NFV Systems (ManSDNNFV'2017), Tokyo, Japan, 2017
Multi-objective multi-constrained QoS Routing in large-scale networks: A genetic algorithm approach - Pham Tran Anh Quang ; Jean-Michel Sanner ; Cedric Morin ; Yassine Hadjadj-Aoul - International Conference on Smart Communications in Network Technologies (SaCoNeT), El Oued, Algérie, 2018
An extended and flexible SDN control plane - Jean-Michel Sanner ; Pierrick Louin ; Yassine Hadjadj-Aoul ; Meryem Ouzzif - International Conference on Smart Communications in Network Technologies (SaCoNeT), El Oued, Algérie, 2018
Virtual Network Function Forwarding Graph Embedding: A genetic algorithm approach - Pham Tran Anh Quang ; Jean-Michel Sanner, Cédric Morin; and Yassine Hadjadj-Aoul - Wiley IJCS, Journal 2019 (to appear)

1.4 Organisation du manuscrit

Ce manuscrit de thèse est organisé de la manière suivante. Nous présentons tout d'abord un état de l'art qui recouvre les trois points suivants: (1) les différentes architectures SDN à base de contrôleurs proposées dans la littérature, (2) les approches adoptées dans la littérature pour le placement de services réseaux, et en particulier pour le placement des contrôleurs de manière détaillée, et (3) une description et une discussion détaillée sur les algorithmes évolutionnaires qui est un des principaux outils que nous avons mis en œuvre.

Nous détaillons ensuite nos contributions dans deux chapitres.

Le premier chapitre présente nos travaux et propositions sur les architectures SDN.

Dans ce premier chapitre, une première proposition présente une architecture SDN distribuée orientée services et dynamique, "Dynamic adaptIve serviCe-drivEn SDN architecture (DICES)".

L'architecture DICES permet de définir et de déployer des services réseaux à la demande en prenant en compte les SLAs attendus pour les services réseaux.

La seconde proposition de ce premier chapitre présente une preuve de concept dont l'objectif est d'illustrer certains principes définis dans l'architecture DICES. Cette preuve de concept illustre en particulier: des services réseaux décomposables en composants

réseaux élémentaires, un plan de contrôle SDN au périmètre plus large et plus flexible que ce que permet OF et une mise à jour des services réseaux en ligne.

Puis vient le deuxième chapitre dans lequel nous présentons nos contributions sur le sujet du placement de services réseaux.

Elles traitent d'abord du placement de contrôleurs, puis du placement de chemins et enfin du placement de chaînes de VNF en s'appuyant sur les algorithmes évolutionnaires.

Pour le placement de contrôleurs, notre contribution propose une approche heuristique basée sur le clustering hiérarchique pour optimiser la latence et la charge et permettant de réaliser un placement dans un délai très rapide même sur des réseaux de tailles significatives.

Puis nous présentons une approche multi-objectifs basée sur les algorithmes évolutionnaires pour optimiser en particulier un objectif de fiabilité. Nous démontrons aussi l'intérêt de cette approche pour réaliser un placement dynamique.

Pour le placement de chemins et de chaînes de VNFs, nous présentons un travail réalisé en collaboration avec Quang Phan Tan post-doc à BCOM, dont l'objectif était d'utiliser là aussi les algorithmes évolutionnaires pour résoudre ces problèmes.

L'approche cherche d'abord à résoudre le problème connu du routage multi-objectifs et multi-contraintes. Elle permet ainsi d'atteindre des performances similaires voire meilleures que des heuristiques spécialisées.

L'approche est ensuite étendue au placement de services réseaux dans une infrastructure virtualisée. Ces services réseaux sont modélisés sous la forme de graphe appelés VNF-FGs.

La conclusion rappelle les principaux points abordés dans les différents chapitres et identifie des axes d'études à développer.

Bibliography

HALPERN, J. M., R. HAAS, A. DORIA, L. DONG, W. WANG, H. M. KHOSRAVI, J. H. SALIM et R. GOPAL. 2010, «Forwarding and Control Element Separation (ForCES) Protocol Specification», RFC 5810, doi:10.17487/RFC5810. URL https://rfc-editor.org/rfc/rfc5810.txt.

MCKEOWN, N., T. ANDERSON, H. BALAKRISHNAN, G. PARULKAR, L. PETERSON, J. REXFORD, S. SHENKER et J. TURNER. 2008, «Openflow: Enabling innovation in campus networks», *SIGCOMM Comput. Commun. Rev.*, vol. 38, nº 2, doi:10.1145/1355734.1355746, p. 69–74, ISSN 0146-4833. URL http://doi.acm.org/10.1145/1355734.1355746.

Chapitre 2

Etat de l'art

" Beware the Jabberwock, my son! The jaws that bite, the claws that catch! Beware the Jubjub bird, and shun The frumious Bandersnatch! "

Lewis Caroll

Sommaire

2.1 Introduction

Dans cet état de l'art, nous brossons un tableau des travaux et publications autour de deux sujets qui ont fait l'objet de nos travaux de thèse: les architectures SDN et le placement de fonctions réseaux dans les infrastructures.

Dans la première partie, nous tentons de dresser un état de l'art des différentes typologies d'architectures SDN généralement distribuées qui ont été imaginées pour mettre en œuvre un réseau SDNisé. Il s'agit toujours d'approches autour de la mise en œuvre d'OF.

Nous avons pour cela essayé d'identifier des publications pertinentes autour des architectures SDN centralisées, distribuées ou hiérarchiques ou encore intégrant une notion de dynamicité. Nous regardons aussi comment les technologies SDN s'insèrent dans le paradigme NFV en plein développement aujourd'hui.

Dans la seconde partie, nous avons considéré un éventail de publications qui traitent du placement de contrôleurs en tenant compte de différentes métriques.

Nous commençons par répertorier des données existantes concernant les métriques et les données chiffrées à prendre en compte pour placer des contrôleurs puis nous identifions certaines règles d'ingénierie de placement.

Ensuite, nous examinons de manière approfondie les approches adoptées pour le placement sur un critère de latence seule, puis sur un critère de latence et charge, puis sur un critère de latence, charge et fiabilité. Nous considérons enfin les approches qui prennent en compte la résilience ou bien le caractère dynamique pour le placement des contrôleurs.

Nous avons ensuite établi un court état de l'art autour du placement de chemins et du placement de VNF-FGs que nous avons aussi abordé dans notre travail. Pour le placement de chemins, nous abordons seulement les heuristiques les plus performantes pour résoudre le problème du routage multi-contraintes et multi-objectifs et un exemple d'approche utilisant les Algorithmes Evolutionnaires Multi-Objectifs (AEMO).

Pour le placement de VNF-FGs, nous analysons deux approches à base de métaheuristiques et une approche basée sur modèle MILP dans une littérature foisonnante sur le sujet.

2.2 Architectures des réseaux SDN

Au début des propositions autour du paradigme SDN, la plupart des travaux envisageaient seulement un contrôleur centralisé pour un domaine donné, ce qui conduisait à une sollicitation importante de la capacité de traitement du contrôleur et conduisait naturellement à des problèmes de mise à l'échelle.

Paradoxalement, une des propositions qui a surgi alors était le concept de contrôleur logiquement centralisé s'appuyant sur une certaine distribution physique de cette fonction comparable à la distribution des fonctions dans les réseaux classiques.

Il est apparu aussi que la fonctionnalité de contrôleur devait être physiquement distribuée pour permettre de gérer des réseaux à grande échelle.

Nous avons donc abordé cet état de l'art en classant les principales propositions d'architecture pour réaliser la centralisation et la distribution du contrôleur SDN. Nous avons aussi abordé les propositions visant à introduire une certaine dynamicité pour répondre aux variations du réseau.

En préambule, nous présentons le paradigme NFV chez les opérateurs TelCo dans lequel s'inscrivent les technologies SDN. En effet ce paradigme est devenu le contexte

principal dans lequel les technologies SDN doivent être mises en œuvre chez les opérateurs.

2.2.1 Architectures virtualisées pour opérateurs Telco

L'idée générale de NFV est de mettre en œuvre des infrastructures composées de serveurs banalisés (Commercial Of The Shelf (COTS)) pour implémenter des fonctions réseaux hébergées dans des machines virtuelles sous forme de VNFs. Notre objectif n'est pas de présenter les différentes approches autour du paradigme NFV, mais simplement de montrer comment SDN, et en l'occurence OF et NFV peuvent s'articuler de manière complémentaire, en particulier:

- Le contrôleur réseau peut être virtualisé et hébergé sur des serveurs NFV, il peut donc s'agir d'une brique réseau virtuelle, d'une VNF,
- Le switch OF (OpenFlow Switch (SOF)) peut-être virtualisé comme par exemple avec Open-VSwitch, [PFAFF et collab., 2015],
- OF est généralement un moyen de déployer la connectivité dans une infrastructure virtualisée, à différents niveaux, à l'intérieur d'une machine virtuelle (Virtual Machine (VM)), dans un serveur pour connecter des machines virtuelles et des Virtual Network Functions Components (VNF-C)s, à l'intérieur d'une VNF, d'un tenant, ou dans l'infrastructure d'hébergement,
- OF est un moyen de réaliser le chaînage de services pour composer des services réseaux en aiguillant les flux entre VNFs ou entre middleboxes virtualisées,
- La brique OF est très répandue dans les datacenters,
- La brique OF est présente dans le Virtual Infrastructure Manager (VIM) pour réaliser la connectivité entre les machines virtuelles, par exemple dans OpenStack,
- La brique OF permet de réaliser une connectivité dynamique et adaptative,
- OF permet de réaliser le concept de slices réseaux présent par exemple dans la 5G.

Sous certains aspects, le paradigme NFV a supplanté l'approche SDN (en particulier OF) en répondant autrement à certains verrous techniques qui initialement étaient adressés par les deux approches. Par exemple, tandis que le SDN essayait de résoudre le verrou technique des middle-Boxes qui se multipliaient dans les infrastructures (fonctions DPI, firewall, NAT ...) en distribuant ces fonctions dans des SOFs améliorés couplés à des contrôleurs plus ou moins distribués, NFV propose plutôt de garder les middleboxes en les virtualisant dans des datacenters, ce qui permet de résoudre les problèmes de coûts et d'évolutivité.

Dans le contexte NFV, le SDN est devenu essentiellement un moyen de gérer la connectivité et la ressource réseau au service de l'ensemble de l'infrastructure et est potentiellement utilisable dans différents blocs fonctionnels:

- dans des tenants ou des infrastructures virtuelles,
- au niveau de l'infrastructure physique elle même qui supporte les tenants,
- dans les VNFs pour relier des VNF-Cs,

- pour chaîner des VNFs et aiguiller les flux de manière à réaliser un service réseau complet.

Les différents blocs fonctionnels peuvent être vus comme organisant une architecture récursive ou hiérarchique de contrôleurs chargés de piloter les ressources réseaux dans les différentes couches [CHATRAS, 2015]. Une tentative d'illustration est présentée dans la figure 2.1. Dans la couche supérieure se trouvent les fonctions Operational Support System (OSS) et Business Support System (BSS) qui regroupent les fonctions supports et métiers de l'opérateur.

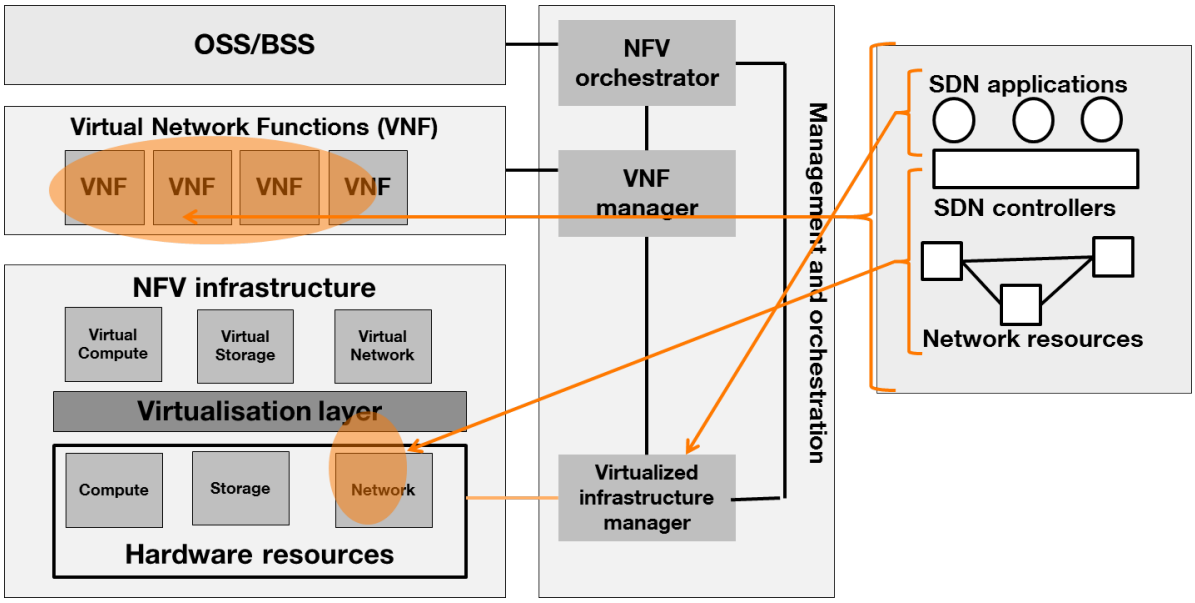


Figure 2.1 – SDN dans NFV

Il faut remarquer cependant que les deux approches SDN et NFV se complètent mais abordent les verrous techniques que constituent l'élasticité et la flexibilité de manière différente, l'une en centralisant les fonctions, l'autre en les virtualisant.

2.2.2 Architectures SDN physiquement centralisées et en clusters

L'organisation des contrôleurs en clusters est un moyen d'améliorer la fiabilité du réseau SDN et de résoudre le problème de défaillance d'un seul point centralisé dans les architectures SDN. La technologie qui organise les clusters est basée sur un processus électif d'un maître parmi des instances de contrôleurs partageant une base de données distribuée.

[JAIN et collab., 2013] ont présenté une implémentation concrète et effective d'OF dans le contexte industriel de la gestion du trafic interne entre les data-centers Google. Les motivations pour la mise en œuvre d'OF étaient principalement la facilité pour la mise en œuvre de nouveaux protocoles, des procédures de tests et de validation plus simples, et enfin et surtout un objectif d'optimisation du trafic entre data-centers avec l'idée de réduire le surdimensionnement des liens par la mise en œuvre d'outils d'ingénierie de trafic.

Pour Google, chaque Data Center constitue un système autonome AS connecté au réseau internet. Ils sont quelques dizaines disséminés dans le monde et sont interconnectés entre eux via le cœur de réseau B4 d'interconnexion. Ce réseau prend la forme

d'un autre système autonome AS qui utilise traditionnellement les protocoles iBGP et IS-IS pour le routage. Dans cette architecture, les services de routage et d'ingénierie de trafic sont déployés de manières indépendantes afin d'assurer un fonctionnement minimal par des procédures de routage classique en cas de défection du serveur d'ingénierie de trafic. Le schéma 2.2 présente une vue de l'architecture globale.

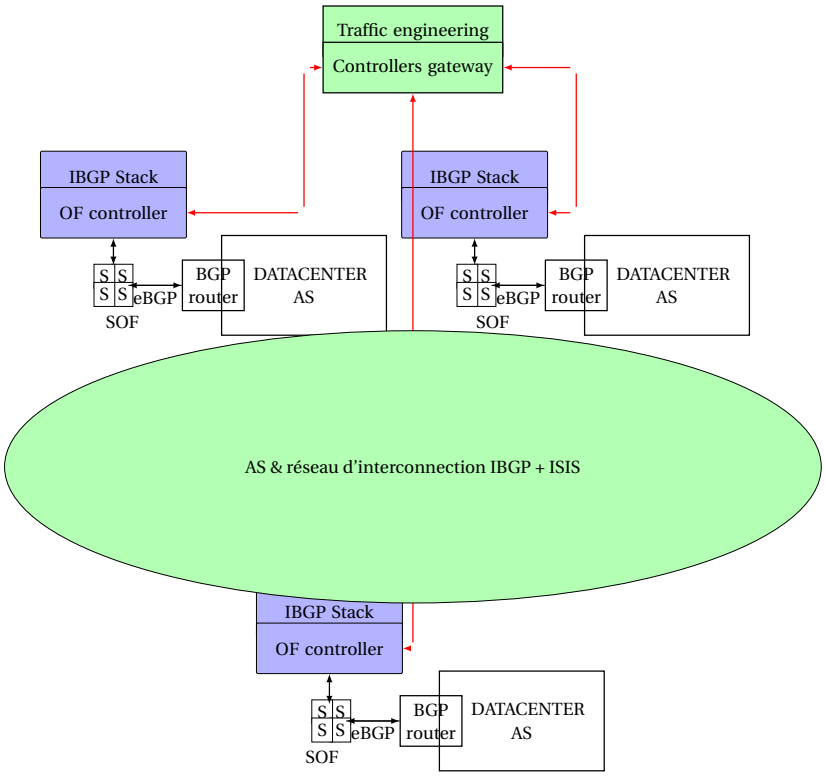


Figure 2.2 – Google B4 network

Au niveau de l'interface entre les data centers et le réseau B4, les routeurs d'interconnexion ont été convertis progressivement en SOFs, et les fonctions de routage ont progressivement été migrées dans des contrôleurs centralisés (rattachés à chaque data center). Ces contrôleurs mettent en œuvre une pile de routage virtuelle Quagga, [JAKMA et LAMPARTER, 2014], qui calcule les routes iBGP et IS-IS, [PUJOLLE et SALVATORI, 2014], qui sont ensuite descendues dans les SOFs. Ces contrôleurs sont organisés en clusters pilotés par un logiciel (PAXO), [PRISCO et LAMPSON, 1997], qui permet d'assurer par un mécanisme d'élection une redondance à chaud en cas de défection d'un contrôleur.

Les SOFs sont des équipements banalisés aux caractéristiques relativement moyennes (buffers, taille des tables de forwarding) qui ont été customisés par Google afin de réduire les coûts. L'optimisation du trafic permet en particulier de pallier les faibles performances des SOFs. De plus, les SOFs ont pu ainsi être rendus conformes aux besoins de Google en terme de contrôle de bas niveau et de nombre d'interfaces.

Pour différencier le trafic de routage du trafic d'ingénierie, les SOFs disposent de deux tables différentes qui sont consultées en parallèle. La table pour l'ingénierie de trafic est prioritaire sur la table de routage.

La solution centralisée d'ingénierie de trafic a été introduite en overlay (en utilisant

des tunnels IP in IP),[PUJOLLE et SALVATORI, 2014], pour optimiser et planifier les flux critiques qui circulent entre les data centers. Cette solution est d'autant plus fiable que des incidents sur la partie ingénierie de trafic provoquent simplement un repli sur le mode de fonctionnement antérieur s'appuyant sur un routage classique. C'est un des avantages procurés par la mise en œuvre d'OF.

Le serveur d'ingéniérie de trafic utilise une abstraction du réseau qui présente chaque datacenter comme un simple nœud du réseau. Le trafic entre deux nœuds peut éventuellement être distribué dans tous les liens du réseau en fonction des politiques adoptées. Le serveur manipule des groupes de flots qui sont une abstraction d'agrégats de trafic correspondant à différentes applications ayant une pondération identique. Ces groupes peuvent être agrégés entre eux quand ils ont la même pondération.

Google utilise une heuristique Greedy non optimale mais annonce un taux d'utilisation des liens proche de 99%. Un taux moyen d'interruption du service divisé par 10 est aussi annoncé grâce à l'organisation des contrôleurs en clusters.

L'algorithme glouton est certainement efficace du fait de la faible complexité du graphe et du type d'applications qu produisent de gros bursts de trafic faciles à planifier).

[YAP et collab., 2017] ont présenté un état d'avancement de la SDNéisation du réseau B2 en bordure d'interconnexions (de peering) avec les fournisseurs d'accès et via lesquels les clients de Google se connectent. Ce réseau transfère un volume significatif de trafic. Le diagramme 2.3 illustre la topologie globale de l'infrastructure Google. Le réseau B4 correspond au cercle central avec les data-centers inter-connectés. Le réseau B2 est le réseau en bordure. Ce réseau qui était jusqu'à présent composé de routeurs traditionnels à haute performance (plusieurs centaines de ports et convoyant des centaines de Gbits de trafic) et d'un plan de contrôle distribué est également en cours de SDNéisation pour réaliser à la fois de l'optimisation fine de trafic et aussi pour gagner en évolutivité d'infrastructure. L'enjeu est d'introduire le SDN au niveau des routeurs de haute capacité

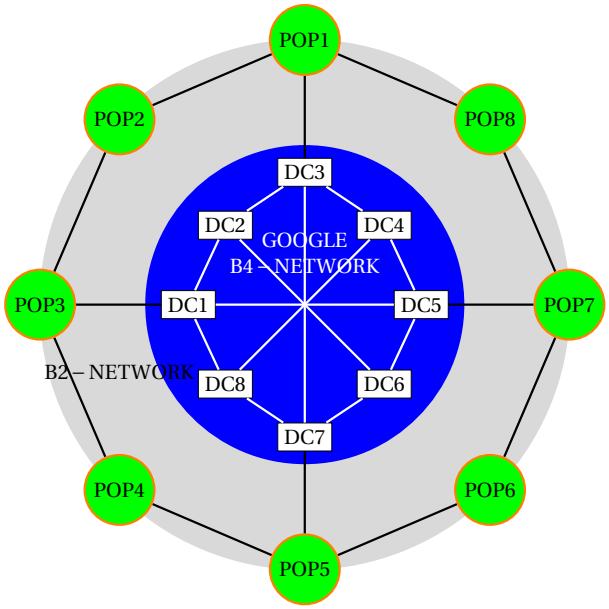


Figure 2.3 – Une illustration de l'infrastructure réseau de Google

qui servent d'interfaces avec l'extérieur. Ces routeurs ont trois fonctions essentielles:

- Traitement et routage sur des centaines de ports d'entrées/sorties hauts débits,
- Gestion de tables de routages à plusieurs centaines de milliers d'entrées,

- Mise en œuvre de listes de contrôle d'accès à large échelle.

Ils sont longs à faire évoluer avec des délais supérieurs à une année pour introduire une nouvelle fonctionnalité. De plus, ils ne permettent pas une ingénierie de trafic fine.

Les deux objectifs de la nouvelle architecture sont, en premier lieu, d'optimiser le trafic vers les utilisateurs et en fonction des applications et, en second lieu, de réduire le temps de cycle pour introduire de nouveaux services.

Pour réaliser cela, de nouveau, une architecture centralisée hiérarchique à deux niveaux de contrôleurs a été adoptée. Elle met d'abord en œuvre un GC (Global Controller) qui bénéficie d'une vision globale et qui est en charge des calculs d'ingénierie permettant d'affiner la manière dont le trafic est écoulé vers les différents ports de sortie et vers les routeurs de peering des clients.

Le GC fournit des décisions d'ingénierie de trafic en lien avec les applications grâce à l'optimisation globale, tandis que les LCs (Local Controllers) sont en charge de descendre les règles de routage et de QoS dans les tables de forwarding des SOFs du plan de données.

Les LCs, par leur localité, permettent aussi une réaction rapide aux défaillances, ce qui permet d'augmenter la fiabilité et la réactivité.

Cette optimisation du trafic qui tient compte des applications, de la bande passante associée aux utilisateurs et de l'occupation des liens n'est pas réalisable via les mécanismes standards de base propres à BGP dans un contexte distribué classique dans lesquels des mécanismes de plus courts chemins sont prépondérants [FEAMSTER et collab., 2003].

[JIN et collab., 2013] proposent **SoftCell**, un exemple d'architecture de réseau mobile basée sur un contrôleur SDN physiquement centralisé pour distribuer les fonctions de base du réseau et rendre celui-ci plus flexible. Cette proposition est plus qu'une simple évolution de l'architecture 4G traditionnelle car les fonctions S-GW et P-GW en périphérie des tunnels Internet et GTP [PUJOLLE et SALVATORI, 2014], sont supprimées en utilisant des mécanismes OF.

Le réseau se compose alors de la fonction de contrôleur SDN, de SOFs d'accès situés à proximité des stations de base chargées d'appliquer des politiques de qualité de service fines, de switchs centraux standards et de passerelles connectées à Internet.

Fournir un service consiste donc à appliquer à un trafic donné une séquence de VNFs à l'aide de techniques d'aiguillage du trafic. Le trafic est identifié à l'aide de balises et d'étiquettes pour fournir différents niveaux d'agrégation et pour résoudre les problèmes de scalabilité dus au nombre élevé de flux à gérer.

Pour gérer les problèmes de fiabilité en cas de défaillance d'un contrôleur, l'architecture SoftCell introduit des agents locaux qui intègrent des répliques de l'état du contrôleur central pour récupérer rapidement l'état des SOFs locaux.

2.2.3 Architectures SDN distribuées à plat et logiquement centralisées

Nous appelons architecture distribuée à plat, une architecture composée de contrôleurs physiquement distribués mais logiquement centralisés.

L'idée est, en distribuant les contrôleurs, de lever les inconvénients d'un seul contrôleur physiquement centralisé tout en conservant les avantages que constituent le fait de disposer d'une vue globale du réseau et d'interagir globalement avec lui.

Les inconvénients d'un contrôleur centralisé sont nombreux. On peut souligner au moins trois aspects:

- Un contrôleur centralisé aura du mal à réagir à des évènements locaux réclamant des traitements locaux, de manière optimale. Il y a donc des compromis à établir entre centralisation et distribution des fonctions.

- Un contrôleur centralisé constitue un goulot d'étranglement, ce qui limite la mise en œuvre de services réseaux performants.

- Un contrôleur centralisé est un "single point of failure".

D'un autre côté, une architecture de contrôleurs distribués introduit de nouveaux verrous techniques associés à la nécessité de garantir la cohérence des informations partagée par les contrôleurs afin de disposer d'une vue unique du réseau. Un protocole est-ouest est nécessaire pour gérer les échanges entre contrôleurs. De plus, le degré de distribution physique peut varier énormément suivant les approches.

[SCHMID et SUOMELA, 2013] analysent la problématique de la localité dans un réseau SDN en fonction des applications réseau mises en œuvre.

La proposition vise à montrer que dans un réseau SDN distribué, on peut souvent définir une composante locale pour les applications réseaux, voire réaliser un traitement purement local plutôt qu'adopter systématiquement une approche centralisée.

L'approche locale permet d'augmenter la réactivité. De plus il n'est pas toujours nécessaire de réaliser une optimisation globale. Enfin, les applications locales peuvent ensuite être coordonnées pour réaliser aussi des fonctions globales sur le modèle des approches adoptées pour les algorithmes distribués.

Deux cas d'usage sont étudiés à titre d'exemple, un cas d'usage de calcul d'un spanning tree et un cas d'usage d'établissement de connectivité client vers des serveurs. Ce type d'approche intéressante n'est pas exploitée aujourd'hui dans le contexte du SDN.

[LEVIN et collab., 2012] proposent d'analyser les verrous techniques liés à la distribution des états inter-contrôleurs dans la mise en œuvre d'une architecture SDN distribuée à plat et logiquement centralisée.

L'implémentation d'applications réseaux de manière distribuée peut-être rendue impossible ou infaisable si des incohérences surgissent dans les vues réseaux dont disposent les contrôleurs et peut générer des boucles et des erreurs de routage.

Les auteurs étudient donc d'une part, une stratégie qui privilégie le compromis entre les performances espérées de l'application réseau ciblée par rapport au surplus de trafic nécessaire pour distribuer les différents états de manière cohérente entre les différents contrôleurs (strongly consistent). Ils étudient, d'autre part, une stratégie qui limite le risque d'incohérences dans le partage de l'information (eventually consistent) en réduisant le surplus de trafic et qui privilégie la complexité de la logique applicative en la rendant capable de prendre en compte certains états du réseau. Le cas d'usage considéré est un load balancer.

[PANDA et collab., 2013a] abordent la question des contrôleurs dans des architectures SDN distribuées à plat, et logiquement centralisées du point de vue du théorème CAP (Consistency, Availability, Partition Tolerance), [GILBERT et LYNCH, 2012], utilisé dans les bases de données distribuées. Pour rappel, le théorème énonce que l'on ne peut réaliser simultanément les trois propriétés: cohérence atomique, disponibilité et tolérance au partitionnement. Dans le cas du réseau SDN et d'un plan de contrôle séparé (outband), les auteurs s'intéressent à trois propriétés présentées comme équivalentes:

- La propriété de disponibilité désigne le fait de pouvoir adresser un message à un SOF même en cas de partitionnement du réseau.

- La propriété de cohérence ou d'exactitude (correctness) désigne le fait de pouvoir implémenter correctement une politique réseau dans l'infrastructure réseau.

- la propriété de tolérance au partitionnement exprime le fait que le réseau doit rester fonctionnel même en cas de partitionnement notamment du réseau de contrôle du à une défaillance d'un nœud ou d'un lien.

A partir d'un modèle très simple de réseau SDN distribué équivalent à un système distribué de transmission de messages asynchrones, les auteurs montrent que dans un réseau SDN, il n'est pas possible de réaliser simultanément ces trois propriétés mais seulement deux d'entre elles. Ils déclinent ensuite ces résultats sur 3 cas d'usage, un cas d'usage d'isolation, un cas d'usage de middle box et un cas d'usage d'ingénierie de trafic.

Ils en tirent ensuite quelques conclusions en terme d'implémentation. Le principal résultat est que si l'on tolère le partitionnement du réseau du fait de pannes, il n'est pas possible de garantir la fiabilité de la mise en œuvre des politiques réseaux.

[KOPONEN et collab., 2010] ont proposé ONIX qui nous semble la première véritable tentative d'architecture SDN distribuée sur réseaux hétérogènes (du WAN au datacenter).

Au-dessus du réseau lui-même se trouve l'infrastructure de connectivité pour organiser la communication entre les instances ONIX et les équipements sur la base d'un protocole de routage standard. Les instances ONIX sont hébergées dans des serveurs.

Au-dessus d'ONIX se trouve comme dans une architecture SDN standard, la logique de contrôle des applications. La principale composante d'ONIX est la NIB (Network Information Base). La NIB est une base de données distribuée qui fournit des API Nord pour les applications et des API Sud pour le déploiement des services réseau. La NIB maintient une vue actualisée du réseau disponible pour la logique applicative au niveau des différentes instances.

Les API sont conçues pour garantir l'accès atomique aux ressources et la synchronisation entre elles si nécessaire. Des éléments peuvent être agrégés et vus comme un seul pour fournir une vue plus simple aux autres instances ONIX.

C'est la logique de contrôle de l'application qui permet de garantir la cohérence. Elle utilise pour cela deux bases de données distribuées, l'une pour assurer une cohérence forte et qui assure la persistance de l'information et l'autre pour une cohérence plus faible qui garantit la disponibilité du service [PANDA et collab., 2013b]. La première base de données stocke des informations durables comme la topologie du réseau, l'autre stocke des informations plus volatiles comme les métriques d'utilisation des liens. Les API fournies aux applications offrent le choix d'adopter un compromis et de privilégier soit des taux de mise à jour élevés et une bonne disponibilité, soit une cohérence plus forte et une meilleure persistance.

[TOOTOONCHIAN et GANJALI, 2010] présentent **Hyperflow** qui peut-être considérée comme une proposition de référence d'une architecture physiquement distribuée et logiquement centralisée. Les contrôleurs contrôlent différents domaines et sont interconnectés entre eux par une interface est-ouest et via une base de données distribuée qui met à disposition différents canaux de communication pour mettre à jour les états des contrôleurs. Chaque contrôleur gère les SOFs qui lui sont connectés et intervient indirectement dans la gestion de l'ensemble du réseau.

L'architecture est illustrée avec des contrôleurs NOX [PALIWAL et collab., 2018] et des SOFs dans un environnement Mininet et une application de base de données distribuée au-dessus des contrôleurs pour synchroniser les vues réseau entre elles via un canal de contrôle utilisant un service de publication de messages. L'application de base de données et de publication de messages permet grâce à ses propriétés intrinsèques de :

- garantir les événements qui doivent être délivrés à chaque contrôleur (en utilisant des modalités de stockage persistantes dans la base de données distribuée),

- fournir une propriété de résilience contre le partitionnement du réseau et les pannes (c'est-à-dire que le réseau continue d'être fonctionnel en cas de partitionnement et le redevient pleinement quand le partitionnement est résolu.

- minimiser le trafic entre les contrôleurs pour propager des événements vers les SOFs.

La résilience est garantie par la capacité de l'application à se resynchroniser après une défaillance du canal de publication/abonnement. Les configurations locales ou les requêtes vers les SOFs sont toujours gérées par les contrôleurs locaux en charge des SOFs correspondants.

L'ordre de grandeur donné pour maintenir la cohérence entre contrôleurs est d'environ 1000 événements par seconde. Cette limitation est principalement une limitation du canal de publication/abonnement de la base de données distribuée.

[PHEMIUS et collab., 2014] présentent une architecture distribuée à plat inter-domaines illustrée par un cas d'usage de perturbation de la topologie inter-domaines. Comme pour hyperflow, l'application distribuée s'exécute au-dessus de la couche des contrôleurs qui est composée cette fois-ci d'entités contrôleurs floodlight. Le protocole de communication inter-contrôleur est basé sur AMQP [1] et RabbitMQ [DOBBELAERE et SHEYKH ESMAILI, 2017] pour composer un système de bus de messages publication/abonnement avec un AMQP client-serveur sur chaque contrôleur. Chaque contrôleur met en place quatre agents chargés de la surveillance, de l'accessibilité, de la connectivité et de la réservation des ressources. Dans le cas d'utilisation présenté, les agents sont combinés pour reconfigurer dynamiquement le réseau en cas de surcharge des communications de contrôle afin d'adapter le trafic à un plan de contrôle aux caractéristiques variables en commutant par exemple le trafic vers une liaison alternative.

2.2.4 Architectures SDN hiérarchiques

Différents travaux proposent une architecture hiérarchique pour adresser le problème de la distribution des contrôleurs et pallier la complexité d'une architecture à plat. Le principe consiste à définir deux niveaux de hiérarchie : un niveau local responsable de la gestion d'un domaine et un niveau supérieur pour gérer la communication entre les contrôleurs locaux et disposer d'une vue globale. Les réseaux B2 et B4 de Google peuvent être vus ainsi, bien que le niveau le plus élevé ne relève que de l'ingénierie de trafic.

[HASSAS YEGANEH et GANJALI, 2012] proposent **Kandoo**, un ensemble organisé hiérarchiquement de contrôleurs compatibles avec OF. Ils définissent une notion de contrôleurs locaux gérant directement un ensemble de SOFs et un contrôleur racine qui bénéficie d'une vision globale du réseau.

Kandoo a l'avantage principal d'éviter la modification ou l'extension des SOFs.

Bien que le document présente des résultats impressionnants avec 90% des événements traités par les contrôleurs locaux, il est à noter que le champ d'application de cette solution est limité aux environnements où les décisions locales sont prédominantes comme les data-centers.

[FU et collab., 2014] présentent **Orion**, un plan de contrôle hiérarchique hybride qui, comme Kandoo, divise le plan de contrôle en un plan de contrôle local et un plan de contrôle global. Afin d'aborder les réseaux à grande échelle, ils introduisent la notion de sous-domaine, divisé en zones. Une des principales différences avec l'architecture de Kandoo

[1] Advanced Message Queuing Protocol

réside dans le rôle de chaque plan de contrôle. Dans Orion, les contrôleurs locaux gèrent une zone entière et proposent une vue abstraite de leur zone à un contrôleur responsable de zone

Une autre différence avec Kandoo est la distribution intrinsèque des contrôleurs de domaine. Alors que Kandoo plaide pour un contrôleur global "logiquement" centralisé, Orion a conçu sa couche contrôleur de domaines avec plusieurs contrôleurs globaux qui interagissent pour la gestion inter-domaines via un protocole distribué.

[SANTOS et collab., 2014] proposent **D-SDN** avec l'idée d'un contrôleur responsable de la gestion d'un sous-domaine du réseau et en interaction avec un contrôleur maître. D-SDN donne un aperçu général d'un concept d'architecture D-SDN. Le document se concentre ensuite sur les questions de sécurité pour la délégation de contrôle entre le contrôleur principal et un contrôleur secondaire. La sécurité des communications entre les contrôleurs avec une hiérarchie à deux niveaux ainsi que leur tolérance aux pannes ont également été étudiées.

2.2.5 Architectures SDN flexibles et adaptatives

[DIXIT et collab., 2014] proposent une architecture distribuée à plat capable d'adapter dynamiquement les pools de contrôleurs et les SOFs associés aux conditions de charge du réseau.

Selon la charge du réseau, la fréquence des paquets envoyés aux contrôleurs varie et les contrôleurs peuvent être sous-chargés ou surchargés. En cas de variation de charge, les pools de contrôleurs peuvent être réduits ou augmentés, et les SOF réaffectés.

L'architecture à plat proposée repose sur une base de données distribuée qui assure une cohérence forte et une bonne communication entre les entités. Les contrôleurs fournissent des informations de mesure de charge pour alimenter un module de décision sur l'interface nord des contrôleurs et pour gérer les décisions d'équilibrage de charge.

[SHAKSHUKI et collab., 2015] décrivent une architecture distribuée à plat flexible. La flexibilité est obtenue en utilisant des instances de contrôleurs virtuels qui peuvent être déployées à la demande sur du matériel COTS fonction des conditions réseaux. Les résultats expérimentaux sont limités à la démonstration de l'intérêt d'une architecture distribuée pour gérer des conditions de charges élevées du réseau, tandis que la propriété de flexibilité n'est pas réellement illustrée.

[TUNCER et collab., 2015] proposent une architecture de management et de contrôle distribuée d'un réseau SDN composées de Local Managers (LM) et de Local Controllers (LC) qui forment des plans de contrôle et de management logiques séparés.

Chaque LC est associé à un ou plusieurs LMs qui sont en charge de la logique distribuée des applications réseaux.

L'architecture proposée permet de gérer des scénarios statiques ou dynamiques de gestion de réseaux. Le déploiement et le placement des LCs et des LMs dépend de la topologie réseau ou du type d'applications supportées et aussi du degré de distribution recherché. Un module de management central est en charge des opérations de management à long terme tout en agrégeant des informations provenant des LCs et des LMs et dispose en particulier d'une vue globale du réseau.

Un LM est constitué d'un module de monitoring, d'un module de management des applications et d'un orchestrateur d'applications. Un LC est constitué d'un module de stockage qui permet de stocker des structures de données locales, d'un module de planification et d'un module d'exécution. Un algorithme de placement des LCs est également présenté. Il est décrit dans la section de l'état de l'art concernant le placement des con-

trôleurs en 2.3.1. Enfin deux cas d'usage d'applications réseaux sont illustrés de manière théorique: un cas d'usage de load balancing et un cas d'usage d'ingénierie de trafic.

2.2.6 Data Planes programmables

Comme nous l'avons mentionné, OF présente des limitations intrinsèques. Il y a des inconvénients significatifs à déplacer toute la logique de contrôle au niveau d'un point central avec la mise en œuvre de contrôleurs centralisés. Prendre toutes les décisions sur un point distant introduit des problèmes de délais de réaction et aussi des problèmes d'évolutivité des services.

Pour optimiser le fonctionnement d'OF, différentes architectures, distribuées, hiérarchique et même dynamiques ont été suggérées. Globalement, ces solutions ambitieuses héritent des défauts d'OF: trop de centralisations, des fonctionnalités réduites dues à l'absence de programmabilité du nœud réseau. La fourniture de services réseaux à la demande reste donc difficile à mettre en œuvre et la mise en œuvre de décisions locales est limitée.

L'idée d'enrichir les SOFs, dépourvus d'intelligence, avec un traitement des flux via une machine à états, a été aussi proposée par exemple par [BIANCHI et collab., 2014]. Cela permet d'enrichir le spectre des fonctionnalités au niveau du SOF. Cependant, même si le concept généralise le système des règles SOF et décharge le contrôleur central, le type de fonctions supportées reste limité à des fonctions assez simples.

Pour aller plus loin dans la complexité et la programmabilité des fonctions supportées localement, diverses contributions proposent de construire un plan de données programmable.

[BOSSHART et collab., 2014] ont proposé le langage **P4** pour accroître la flexibilité d'OF. P4, permet de créer des plugins, qui peuvent être compilés dans un SOF reconfigurable en terme de protocole de contrôle.

L'idée générale est que plutôt que d'enrichir à l'infini le protocole OF, il est préférable d'introduire une flexibilité dans la programmation du SOF de manière à réinterpréter les messages du protocole. Une illustration est donnée avec le diagramme 2.4. Le switch est conçu et structuré pour accueillir des protocoles variés qui sont déterminés par programmation.

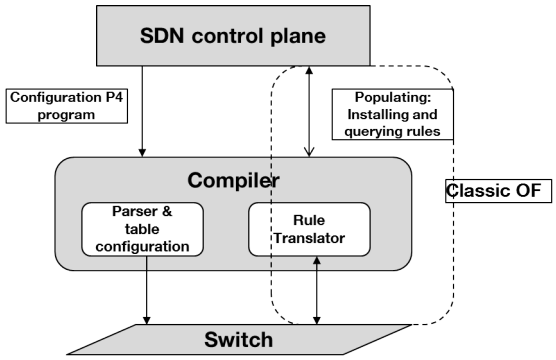


Figure 2.4 – P4

Le SOF devient de fait indépendant du protocole car il peut être programmé pour traiter de nouveaux protocoles éventuels.

Le modèle adopté pour le SOF P4 s'inspire d'OF mais est plus flexible: le parser est programmable permettant de traiter différents en-têtes, les tables de flots peuvent être

agencées par programmation pour effectuer des traitements en série ou en parallèle. Finalement un programme P4 contient de manière déclarative:

- une spécification des en-têtes de protocoles,
- une logique de tables de flots et les actions associées,
- une spécification du parser sous la forme d'un modèle à base de graphe,
- des actions composées à partir de primitives de base, un programme de contrôle qui définit l'ordre de mise en œuvre des tables et actions associées à un paquet.

La re-programmation du SOF n'est pas réalisable de manière dynamique (pendant des traitements en cours), mais les auteurs du papier laissent ouverte cette possibilité éventuellement de manière partielle.

[JOUET et PEZAROS, 2017] proposent la plate-forme **BPFabric** pour programmer et monitorer le plan de données de façon centralisée en utilisant le filtre de paquets de Berkeley étendu (eBPF) des OS Linux associé aux APIs permettant sa programmation. Berkeley Packet Filter (BPF) est un jeu d'instructions et un interpréteur logé dans le noyau Linux.

Le jeu d'instructions tourne dans une machine virtuelle dans le noyau et est appliqué à l'entrée et à la sortie du data path du noyau. C'est aussi un mécanisme générique et sécurisé pour le traitement des paquets.

La toolchain de BPF est composée de l'application qui se trouve dans l'espace utilisateur et du compilateur qui produit un bytecode chargé par un loader dans l'interpréteur. Par rapport à BPF, eBPF dispose d'un jeu d'instructions étendu avec plusieurs améliorations, enfin, BCC est une boîte à outils qui rend les programmes eBPF "plus faciles" à écrire et à manipuler, en C, Python, P4 voire d'autres langages de plus haut niveau.

[JOUET et PEZAROS, 2017] proposent aussi **HLL**, un langage de programmation de haut niveau qui permet de programmer le switch pour réaliser des fonctions diverses à partir de eBPF. L'architecture est illustrée dans le diagramme 2.5. Elle a pour objectif d'augmenter la programmabilité d'un switch architecturé sur un OS Linux dans une machine virtuelle ou physique en s'appuyant sur eBPF.

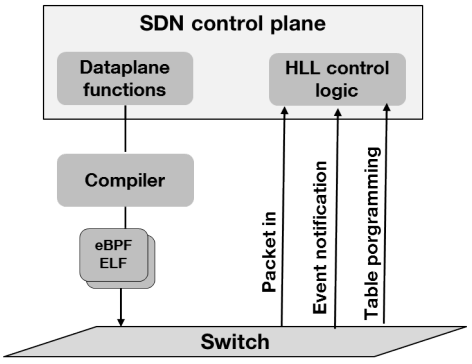


Figure 2.5 – BPF Fabric

Deux implémentations sont illustrées sur une plateforme netFPGA, la première de référence, la seconde mettant en œuvre un accélérateur DPDK. Enfin, afin de faciliter l'interaction avec ce framework de bas niveau, les auteurs ont proposé l'utilisation d'un langage de haut niveau, similaire à P4, et de le compiler dans eBPF[2].

[2]IOvisor Project. https://www.iovisor.org/.

Il faut noter que la proposition ne permet pas de faire une mise à jour dynamique de la configuration du switch pendant des traitements de flux (reconfiguration à chaud).

[KOHLER et collab., 2000] proposent **Click**. Click est un routeur logiciel exécutable dans un environnement Linux. Par ailleurs, il est reconfigurable et modifiable en ligne et hors ligne. Il est décomposable en briques aussi élémentaires que possible qui peuvent être assemblées en réalisant leur composition sous forme d'un graphe orienté pour être exécutées dans un moteur Click hébergé dans un environnement Linux. L'originalité et la plus-value de Click résident dans sa capacité de décomposition fine. Au delà de la notion de routeur logiciel, l'environnement Click permet de réaliser de manière flexible des middle-boxes logicielles supportant une grande variété de fonctions réseaux: analyse paquets, firewall, gateways en bordure d'infrastructure, etc.

[MARTINS et collab., 2014] ont proposé **ClickOS**, un projet Unikernel adapté pour Click. Les Unikernels sont des images d'OS, avec un espace machine à une seule adresse et construites à partir des librairies des OS cibles. ClickOS est aussi un système d'exploitation minimaliste, sur mesure et virtualisé pour l'exécution de middleboxes basées sur Click.

Nec-Labs a apporté quelques modifications à Click pour le faire fonctionner dans l'environnement MiniOS.

Les analyses de performance montrent que les machines virtuelles ClickOS sont très petites, avec une taille de l'ordre de 5 Moctets. Elles démarrent très rapidement dans un délai inférieur à 20 ms, et introduisent peu de latence, de l'ordre de 45 µs. Sur un serveur standard on peut avoir une capacité de traitement de l'ordre de 10 Goctets par seconde.

[LAUFER et collab., 2016] construisent une pile TCP dans l'environnement Click de manière à assurer les fonctions de base nécessaires pour gérer de manière dynamique des sessions TCP sans définir de manière "statique" un élément gérant une session pour chaque connexion TCP. Ainsi, des applications faisant appel aux couches supérieures d'IP peuvent dialoguer directement avec la couche TCP Click sans passer par la pile TCP du système d'exploitation.

[BARACH et collab., 2018] présentent un état de l'art de différentes approches dérivées de Click qui permettent d'accélérer le traitement des paquets. Ces approches sont basées entre autre sur la parallélisation du traitement des paquets sur des systèmes multi-cœurs.

[DA CRUZ MARCUZZO et collab., 2017] présentent un environnement adapté à Click basé sur l'unikernel OSv offrant deux modes de virtualisation (Paravirtualisation - PV, et Hardware assisted virtualisation - HVM) permettant des performances compatibles avec les exigences de middle-boxes standards. L'environnement met en œuvre l'accélération DPDK pour accélérer le traitement des paquets. La démarche est comparable à ClickOS et se calque sur le modèle des fonctions réseaux virtualisées NFV.

[BARBETTE et collab., 2015] proposent **FAstClick** qui met en œuvre les frameworks DPDK et SNAP pour accélérer le traitement des paquets, tandis que [SUN et RICCI, 2013] délègue et parallélise le traitement des fonctions Click à des cartes Graphic Processor Unit (GPU) en s'appuyant sur le framework SNAP.

[KIM et collab., 2012] proposent **DoubleClick** avec diverses améliorations de l'architecture Click en introduisant des traitements batchs sur les entrées-sorties en particulier pour améliorer le traitement des paquets.

[LI et collab., 2016] proposent **NPClick**, une version de Click basée sur une plateforme d'accélération FPGA tout en conservant la programmabilité haut niveau de Click. Enfin, [SHAH et collab., 2004] adoptent une approche similaire basée sur les networks processors.

Pour terminer, [INTEL, 2017] propose **VPP** (Vector Packet Processing) issu d'un projet

open-source Linux porté par Cisco. Il s'agit d'un routeur logiciel complet open-source. VPP est vu comme une application dans le user space linux. Cette solution combine la mise en œuvre de différents mécanismes de bas niveau permettant de traiter une suite de paquets (un vecteur de paquets) dans un même intervalle de temps dans une machine standard de type serveur COTS ou dans une machine virtuelle en bénéficiant de traitements parallélisés. Elle permet de réaliser ainsi des fonctions réseaux complexes avec un traitement très rapide.

Le traitement parallèle est rendu possible par une modélisation du traitement des paquets sous forme de graphe orienté, généralement un arbre, dans laquelle certaines actions sur les paquets peuvent être effectuées simultanément grâce à un algorithme qui distribue un vecteur de paquets dans les nœuds de traitement du graphe, éventuellement en segmentant le vecteur en sous vecteurs de paquets.

La taille du vecteur de paquets résulte d'un compromis entre latence et vitesse de traitement.

Chaque nœud du graphe expose des APIs de contrôle qui peuvent être mises en œuvre via des commandes CLI ou par une application de messages distribuée.

Le graphe peut être enrichi dynamiquement sans passer par une étape de recompilation en rajoutant des nœuds portant des fonctions supplémentaires de traitement. D'autre part, [CHOI et collab., 2017] proposent une extension utilisant le langage P4 de [BOSSHART et collab., 2014] pour créer des plug-ins qui peuvent être dynamiquement installés dans un switch virtuel.

Comparativement à Click, VPP est une solution bien plus aboutie d'un point de vue industriel. La plus-value de Click réside cependant dans l'originalité de l'approche de décomposition de services réseaux en composants élémentaires aussi simple que possible qui peuvent être assemblés via un langage qui modélise un graphe.

Conclusion sur les architectures SDN

Parmi les différentes propositions que nous avons analysées, le cas d'usage Google semble le plus proche d'un cas d'usage opérateur. Il ne peut cependant pas être transposé dans un contexte Telco dans la mesure où les problématiques d'ingénierie de trafic diffèrent fortement. Cependant, des problématiques comme l'optimisation du peering, l'interconnexion de datacenters avec le réseau d'un opérateur ou la mise en œuvre d'infrastructures virtualisées de type NG-POP ou NFV peuvent constituer des opportunités très intéressantes. En effet, dans ce contexte on peut imaginer de gros échanges de données pour certaines applications. Au moins quatre conditions sont nécessaires:

- pouvoir mettre en œuvre une stratégie permettant de réduire la granularité des flux, en traitant plutôt des agrégats et disposer ainsi d'un trafic dont les caractéristiques se prêtent bien à une politique d'ingénierie de trafic,
- disposer de graphes d'interconnexion décrivant la topologie qui permettent de dérouler des algorithmes d'ingénierie de trafic,
- disposer d'une connaissance fine et temps réel de l'écoulement du trafic,
- disposer d'outils d'ingéniérie de trafic de type PCE.

Pour ce qui concerne les approches de dataplanes programmables, elles semblent permettre d'offrir les outils nécessaires pour réaliser ce que OF ne permet pas, à savoir

un plan de contrôle SDN suffisamment flexible et riche pour bénéficier à la fois de la centralisation sans pénaliser le spectre des fonctionnalités qui peuvent être mises en œuvre dans le plan de données.

2.3 Placement de services réseaux

Dans cette section, nous présentons un état de l'art autour des techniques permettant de résoudre différents problèmes de placement dans le contexte des réseaux des opérateurs Telco. Il s'agit:

- des différentes approches adoptées pour traiter du placement de contrôleurs SDN,
- de techniques permettant de placer de manière optimale des chemins dans un réseau,
- d'approches pour le traitement du placement de chaînes de VNF qui implémentent des services réseaux.

L'état de l'art est centré sur le placement de contrôleurs SDN qui a constitué une bonne partie du travail de thèse. Néanmoins, nous abordons la problématique du placement de chemins et de chaînes de VNF dans un souci de généralisation. L'objectif est de définir les principes de conception d'un outil générique de résolution de problèmes de placement de services réseaux.

Ce qui est formulé ici comme des problèmes de placement, peut être aussi bien formulé comme des problèmes d'allocation de ressources. En effet placer des fonctions dans un réseau consiste à allouer des ressources pour ces fonctions. Dans la notion de placement, il y a cependant l'idée de localisation géographique, mais les ressources sont généralement géographiquement localisées.

Les algorithmes proposés pour le placement de services réseaux peuvent être mis en œuvre dans différents contextes. Pour le routage, c'est-à-dire le placement de chemins, le concept de PCE décrit par [FARREL et collab., 2006] est mis en œuvre dans les réseaux MPLS et permet d'appliquer des algorithmes globaux pour réaliser de l'ingénierie de trafic. Il peut être étendu au SDN.

Pour le placement de contrôleurs ou de fonctions réseaux VNF diverses hébergées dans des machines virtuelles, l'outil Watcher[3] associé à Openstack [SHRIVASTWA et collab., 2016] permet d'héberger des algorithmes de placement en bénéficiant de données de monitoring.

Enfin, le projet Open Network Automation Platform Optimisation Framework (ONAP-OF) [4] de la plateforme Open Network Automation Platform (ONAP) vise à développer un framework d'optimisation pour gérer le déploiement de services réseaux modélisés par des VNF-FG dans une infrastructure virtualisée.

2.3.1 Placement de contrôleurs SDN

Le problème ancien du placement de serveurs centralisés dans un réseau constitue une très bonne introduction à des problématiques plus récentes comme celles du placement de contrôleurs ou de fonctions réseaux virtualisées. Ces serveurs doivent en effet pouvoir supporter des fonctions variées à la fois distribuées et centralisées, qu'il s'agisse de base de données, de fonctions de routage ou d'autres applications. L'adaptation du degré de centralisation ou de distribution permet d'optimiser les coûts et d'améliorer la fiabilité.

[BAR-ILAN et PELEG, 1991] présentent les problèmes d'optimisation associés à la problématique du placement de serveurs dans un réseau. Ils considèrent à la fois un critère

[3] WWW.DEVCONF.RU [2015]

[4] HTTPS://WIKI.ONAP.ORG/DISPLAY/ SISUKAPALLI [2017]

de latence entre les serveurs et les nœuds du réseau qu'ils desservent ainsi qu'un critère d'équilibre de charge entre les différents serveurs. Des heuristiques sont proposées et leur coefficients d'approximation donnés. En fixant un nombre maximum de nœuds attachés à chaque serveur, l'objectif est :

- soit de placer un nombre k de serveurs en minimisant la latence entre les serveurs et les nœuds qu'ils servent (k-center problem: [HOCHBAUM et SHMOYS, 1984],
- soit, en fixant une borne maximale ρ pour la latence entre les serveurs et les nœuds servis, de minimiser le nombre de serveurs (ρ dominating set problem: [LOVÁSZ, 1975].

Les deux problèmes peuvent être vus comme duaux l'un de l'autre et sont tous les deux NP difficiles. Le problème "k center" peut être résolu avec un algorithme en temps polynomial avec un coefficient d'approximation de 2, d'après [HOCHBAUM et SHMOYS, 1984].

Le "ρ dominating set problem" peut être résolu par un algorithme glouton en temps polynomial avec un taux d'approximation en $\log n+1$ d'après [LOVÁSZ, 1975] où n désigne le nombre de nœuds du réseau.

Le schéma 2.6 présente grossièrement les points saillants d'une architecture SDN distribuée. On distingue au dessus le plan des applications réseaux. Au milieu, on trouve le plan de contrôle dans lequel plusieurs contrôleurs peuvent collaborer et échanger des informations via un réseau de type spanning tree généralement. Enfin, on distingue l'infrastructure physique ou virtuelle de traitement des paquets c'est-à-dire le plan de données dans les SOFs qui transmettent les paquets. Les différents plans sont inter-connectés via des API nord et sud. L'API nord permet aux applications de dialoguer avec le/les contrôleurs. Les API sud permettent aux contrôleurs de dialoguer avec l'infrastructure physique pour descendre des règles ou récupérer des informations susceptibles d'activer de nouvelle règles.

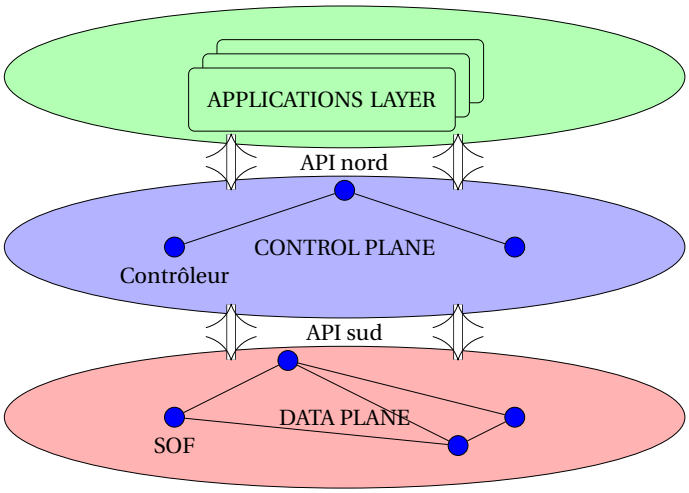


Figure 2.6 – Architecture SDN

Le but du placement des contrôleurs SDN est de placer ceux-ci de manière à fournir le plus efficacement possible des services réseau. C'est un problème clé dans les réseaux SDN. C'est le cas avec un réseau SDN doté d'un seul contrôleur car il faut dans ce cas le placer de manière aussi optimale que possible. Cet objectif est également recherché lorsque plusieurs contrôleurs sont distribués dans le réseau à différentes échelles et selon

différentes architectures (architectures plates ou hiérarchiques, fortement ou faiblement distribuées, ou avec différents degrés de centralisation logique...) comme décrit dans l'état de l'art sur les architectures SDN au chapitre **2.2**.

En effet, la QoS des services réseau qui sont pilotés de manière centralisée ou partiellement centralisée va dépendre de la stratégie de placement des contrôleurs dans le réseau et de leur localisation finale. Cette localisation déterminera des métriques comme la latence entre contrôleurs et les SOFs, ou la latence entre contrôleurs ou encore la charge de trafic des contrôleurs et du plan de contrôle, la sensibilité aux pannes réseaux de l'architecture réseau SDN.

Du point de vue des services réseau, le placement peut influencer par exemple le temps de convergence dans le cas d'un service de routage centralisé au niveau des contrôleurs ou bien le délai de récupération d'une route du fait du temps nécessaire pour calculer un itinéraire alternatif en cas de ré-acheminement rapide (fast rerouting).

Enfin, le placement des contrôleurs SDN pourrait influencer la qualité et la cohérence de la vue réseau qu'ils doivent partager. Sa cohérence pourrait être limitée par la latence et la charge du plan de contrôle ou même de données (plan de contrôle inband), ce qui pourrait affecter la fiabilité du réseau SDN.

Métriques pour le placement

Avant d'exposer différents exemples de publication qui abordent la question du placement de contrôleurs, nous avons recherché des données sur les métriques à prendre en compte pour le réaliser. Nous concluons par un tableau récapitulatif des différentes métriques.

Capacité de traitement [CURTIS et collab., 2011] soulignent dans leur travail qui se situe dans le contexte du datacenter, les limites de la centralisation du plan de contrôle de l'approche SDN ainsi que les limitations propres aux SOFs. Ils citent en particulier la taille des tables de flots nécessaires pour assurer une vue globale du réseau par le biais d'informations statistiques qui remontent des SOFs et le temps de traitement du contrôle pour l'ensemble des flux qui ne font pas appel aux ressources CPU propres au plan de données.

L'ordre de grandeur des capacités de traitement du plan de contrôle est très faible par rapport à celui du plan de données. Les grandeurs évaluées sont de l'ordre de 80 Mbits/s contre 300 Gbits/s. Par conséquent, les traitements dans le plan de contrôle sont susceptibles d'introduire des retards dans les traitements du plan de données.

Charge des contrôleurs [YAO et collab., 2014] donnent des éléments sur la charge des contrôleurs. Le taux de flots générés dans l'ensemble d'un réseau WAN est estimé à $400K \cdot N/s$ où N est le nombre de SOFs du réseau. Ce taux est estimé à partir des mesures effectuées dans le réseau européen CERNET.

Trafic de contrôle Il peut être difficile d'estimer la charge en terme de trafic de contrôle qui dépend en particulier de la variabilité du trafic ou des applications et des besoins qu'elles génèrent en terme d'établissement de nouveaux flots.

[DIXIT et collab., 2014] fournissent les données suivantes concernant le trafic de contrôle dans un datacenter. Des variations d'amplitudes de 1 à 2 ordres de grandeurs sont indiquées entre les valeurs crêtes et médianes des flux arrivant aux SOFs.

En prenant l'hypothèse d'un datacenter hébergeant 100000 VM avec 32 VM par rack, le débit crête d'arrivée de nouveaux flots peut être de 300 Mbits/s tandis que le débit médian peut être compris entre 105 Mbits/s et 10 Mbits/s.

En supposant une capacité de traitement de 2 Mbits/s par contrôleur, le nombre de contrôleurs doit donc pouvoir varier théoriquement de 5 à 150 pour traiter la charge médiane jusqu'à atteindre la charge crête.

[BENSON et collab., 2010] indiquent aussi que le trafic peut atteindre des pointes de 10^6/s arrivées de demandes de nouveaux flots dans le pire des cas dans un réseau composé de 100 SOFs. Il s'agit là aussi d'un datacenter.

Délais induits au niveau du contrôleur et capacité de traitement Deux métriques sont traditionnellement regardées lorsqu'on veut connaître les performances d'un contrôleur SDN. On cherche d'une part à évaluer dans quel délai le contrôleur peut répondre à des requêtes et installer de nouvelles règles dans les SOFs et d'autre part combien de requêtes le contrôleur peut traiter par seconde.

[TOOTOONCHIAN et collab., 2012] présentent ainsi plusieurs diagrammes donnant des performances comparatives de différents contrôleurs pour ces deux paramètres.

Les paramètres considérés sont au niveau du contrôleur: d'une part, le minimum response time du contrôleur et d'autre part son maximum load response time obtenu lorsqu'il doit traiter un grand nombre de requêtes. Ces délais interviennent dans le délai d'établissement des flots.

Aux délais de traitement introduits au niveau des contrôleurs doivent être ajoutés, les délais introduits au niveau des SOFs qui dépendent des technologies employées ou du type: virtual switchs ou hardware switchs.

[YEGANEH et collab., 2013] donnent des valeurs indicatives de latence pour l'établissement d'une règle de forwarding dans un SOF hardware ou software dans un réseau non chargé. Les faibles performances du SOF hardware sont probablement dues à la limitation des ressources sur ce type de matériel et à la faible qualité de l'implémentation.

Switch	number of flows installed per second	latency for flow activation
OVS	≥ 10000	$\leq 1.0ms$
Harware	≥ 1000	$\leq 10ms$

D'autre part, les délais d'établissement des règles OFs dans le réseau doivent être mis en adéquation avec les topologies en prenant en compte la latence des liens et la quantité de trafic réseau qui génèrent ces requêtes. La quantité de trafic au niveau des contrôleurs et des SOFs peut saturer les files d'attentes et générer des délais supplémentaires.

En résumé le délai d'établissement d'une nouvelle règle OF dépend de la latence au niveau du contrôleur, de la capacité de traitement du contrôleur, de la latence au niveau du SOF, de la capacité de traitement du SOF hardware ou software, de la latence entre le contrôleur et le SOF, de la charge globale du réseau et enfin aussi de la charge en terme de trafic de contrôle.

Une question pour l'ingénierie d'un réseau SDN est donc quelles métriques faut-il prendre en compte lorsqu'un opérateur veut déployer un contrôleur dans un réseau Telco.

Les acteurs d'ONOS proposent dans leur white paper [ONO, 2017] deux métriques:

- Le débit en charge induit aux interfaces nord et sud du contrôleur,

- Le délai de recouvrement en cas de panne d'un élément réseau en cas de modification de la topologie.

Trafic inter-contrôleurs Comme nous le décrivons dans la section 2.2.3 de l' état de l'art, il peut être nécessaire de distribuer plusieurs contrôleurs dans le réseau de manière à élaborer un compromis qui permette de lever les inconvénients d'une architecture trop centralisée. Cette distribution des contrôleurs où les différents contrôleurs contrôlent différents domaines et partagent le réseau offre des avantages. On peut citer sans être exhaustif:

- Une plus grande robustesse et résilience, grâce à des mécanismes de sauvegarde entre les contrôleurs dans un cluster de contrôleurs,
- Une répartition de la charge entre plusieurs contrôleurs et un effet de "load balancing",
- Une meilleure scalabilité dans la mesure ou la charge du réseau peut-être répartie,
- Une latence moins élevée pour descendre des règles du fait d'une certaine localité du contrôleur,
- Une plus grande réactivité aux évènements locaux du fait de la meilleure localité.

Cependant, elle implique de mettre en œuvre un contrôle logiquement centralisé entre les différents contrôleurs. Les contrôleurs doivent pouvoir partager une vue réseau à jour et mise à jour de manière temps réel c'est-à-dire en respectant des contraintes de délai.

Cela nécessite de mettre en œuvre des mécanismes de synchronisations des informations contenues dans les bases de données des différents contrôleurs. Les contrôleurs doivent partager une vue cohérente et synchronisée du réseau. Pour cela, il est nécessaire de mettre en œuvre un protocole de consensus entre contrôleurs. Il existe différents protocoles possibles et différentes manières de procéder.

[MUQADDAS et collab., 2016] étudient le trafic généré par ce genre de protocole, avec une orientation pour le contrôleur ONOS. Ils proposent un modèle linéaire expérimental de calcul de la quantité de trafic qui prend en paramètres le nombre de SOFs dans un domaine contrôlé, le nombre de liens intra-domaines et inter-domaines. Cependant, le modèle est restreint à trois contrôleurs.

Plus généralement, le trafic inter-contrôleurs est composé :

- du trafic nécessaire pour la mise en œuvre du protocole de consensus,
- des informations nécessaires à la connaissance de la topologie du réseau,
- des messages de vérification de la disponibilité des autres contrôleurs (heart-beat messages),
- des messages assurant la coordination et la synchronisation entre les contrôleurs lors de l'installation ou la suppression de nouveaux flots.

Le trafic inter-contrôleurs peut donc être significatif et être handicapant dans certains cas car il peut générer de nouvelles contraintes. En particulier, la nécessité d'assurer un consensus entre les contrôleurs avant de descendre des règles vers les SOFs peut limiter la réactivité pour descendre de nouvelle règles.

[BIANCO et collab., 2016] abordent cette question. La manière dont le protocole assure le consensus soit de manière contraignante, soit de manière flexible joue un rôle. En effet différentes stratégies et solutions peuvent être adoptées.

[PANDA et collab., 2013b] envisagent cette question du point de vue des contraintes induites par le théorème Consistency Availability, Partition tolerance (CAP), tandis que [GILBERT et LYNCH, 2012] l'envisagent dans un contexte réseau SDN pour ce qui concerne la fiabilité.

En conclusion, la problématique du placement des contrôleurs dans un réseau doit prendre en compte un certain nombre de métriques. Nous en indiquons quelques unes dans le tableau 2.1 qui ne prétend pas être exhaustif.

LATENCY	LOAD	RELIABILITY	RESILIENCE
Worse case latency	Controllers load	Controllers failure rates	Mean down time
Average case latency	Switches load	Switches failures rates	Mean recovery time
Maximum cover latency	Control plane traffic matrix	Link failure rate	Maintainability
Application latency	Data plane traffic matrix	Rate of control path loss	Mean time to incident recovery
Inter-switches latency	control plane traffic overflow	nodes and edges Connectivity	Maximum time between failures
Inter-controllers latency	Load balancing between controllers	Number of controllers nodes	number of disjoints paths between nodes
Shortest path latency between nodes			

Table 2.1 – Métriques pour le placement des contrôleurs

En complément des différentes métriques, il faut aussi prendre en compte pour le placement des contrôleurs des contraintes fonctionnelles qui peuvent intervenir. Il s'agit, par exemple, de la capacité des nœuds à accueillir la fonction contrôleur en terme de traitement, mémoire, stockage.

La plupart des travaux considèrent que l'ensemble des nœuds du réseau sont susceptibles de supporter la fonction contrôleurs, mais en réalité, du fait de la virtualisation, il est probable qu'il faille considérer simplement un sous ensemble d'entre eux, répartis dans une partie du réseau. Il peut s'agir par exemple de micro-datacenters en périphérie ou de gros datacenters plus centraux.

Enfin, on peut aussi chercher à optimiser des métriques liées à la fiabilité qui prennent en compte le taux de pannes de nœuds ou de liens ou encore la connectivité de sous graphes du réseau.

Typologies pour le placement de contrôleurs

La problématique du placement de contrôleurs est abordée généralement dans la littérature uniquement du point de vue de la technologie OF avec toutes les limitations

inhérentes à ce protocole que nous rappelons dans le chapitre sur l'architecture. Nous avons adopté dans la suite une classification des publications en fonction des métriques traitées. La classification reste approximative car les approches du sujet sont très touffues et les techniques très diverses. Parfois plusieurs métriques sont regroupées, parfois une seule est considérée isolément.

Un autre axe d'analyse aurait pu être les techniques adoptées pour la résolution du problème posé: MILP, heuristiques, méta-heuristiques, distribuées, mais ces techniques sont souvent mêlées et comparées dans les articles.

Placement des contrôleurs sur un critère de latence seule [HELLER et collab., 2012] étudient le problème de manière qualitative en s'appuyant sur une analyse en force brute. La question posée est le placement optimal de k contrôleurs sur la base de deux métriques.

Il s'agit d'une part de la latence moyenne calculée du plus court chemin vers leurs contrôleurs pour tous les nœuds attachés aux différents contrôleurs.

Il s'agit, d'autre part, de la latence maximale (pire cas) du plus court chemin entre contrôleurs et nœuds pour un nombre donné de contrôleurs dans le réseau.

L'objectif est de minimiser l'une ou l'autre avec des résultats qui peuvent varier suivant la topologie du réseau.

L'impact des deux métriques n'est pas le même: la latence moyenne par exemple risque de cacher des scénarios pathologiques de nœuds isolés très éloignés des autres. Le réseau est donc partagé en différents domaines ou clusters attachés à chaque contrôleur. L'analyse ici est purement qualitative dans le contexte des réseaux WAN et se concentre uniquement sur la latence liée à des contraintes physiques c'est-à-dire essentiellement la longueur des liens et le temps de propagation. Les problèmes d'optimisation associés minimum K-moyennes et minimum K-center [5] sont NP difficiles et peuvent nécessiter des calculs très longs (plusieurs semaines) si l'on cherche à les résoudre en force brute même sur des topologies de tailles modestes de quelques dizaines de nœuds, s'il y a plusieurs contrôleurs. L'analyse montre que beaucoup de topologies WAN (nation wide principalement) pourraient en théorie fonctionner avec un seul contrôleur en optimisant uniquement ce critère de latence et en prenant comme référence des valeurs de réactivités typiques recherchées pour des services réseaux (comme le délai de 50ms pour un fast-rerouting en MPLS).

L'analyse montre aussi que l'augmentation du nombre de contrôleurs n'agit pas proportionnellement à la réduction de la latence et dépend fortement de la topologie. Enfin, les auteurs ne prennent pas en compte la charge des clusters c'est-à-dire le nombre de SOFs pilotés par un contrôleur. Cela peut emmener des configurations très déséquilibrées en fonction de la topologie du réseau, avec un seul contrôleur isolé en périphérie, par exemple.

[WANG et collab., 2016] proposent une analyse expérimentale d'une mise en œuvre de l'heuristique K-moyennes utilisée en clustering pour placer des contrôleurs sur un critère de latence calculée sur la base du plus court chemin, en résolvant ainsi la problématique du temps de calcul en force brute.

Le nombre de contrôleurs est fixé et des itérations combinées de l'algorithme K-moyennes, qui fonctionne par améliorations locales, permettent d'améliorer les résultats par rapport à une simple itération. Il n'y a pas d'évaluation réelle des performances si ce n'est par simple différence avec la version standard de K-moyennes. En particulier il n'y a

[5] les désignations peuvent varier suivant les auteurs

pas de comparaison avec une modélisation MILP qui aurait pourtant été fructueuse dans ce scénario simple.

[HAN et collab., 2016] abordent la problématique de la latence du plan de contrôle avec un point de vue original en s'appuyant sur la théorie de la contrôlabilité des réseaux [LIU et collab., 2011]. A partir de cette approche, [DAI et collab., 2015] introduisent une notion de latence minimum (MCL) pour garantir une contrôlabilité satisfaisante d'un réseau qui évolue dynamiquement. L'idée est que le réseau doit toujours converger vers l'état attendu dans un délai inférieur au MCL. Il faut, par conséquent, trouver le nombre de contrôleurs minimum pour assurer que le réseau peut toujours converger vers l'état visé dans un intervalle de temps donné.

[HAN et collab., 2016] transposent les idées du papier précédent aux réseaux SDN. Dans ce cas, Le MCL est en quelque sorte le temps minimum toléré pour obtenir une réponse d'un nœud à une commande du contrôleur. La matrice de latence est donc établie en multiples de ce temps élémentaire. La limite de ces travaux réside dans les facteurs d'approximation liés aux algorithmes proposés qui impliquent beaucoup de contrôleurs.

[KILLI et collab., 2018] proposent une approche du placement des contrôleurs sur un critère de latence inspirée de la théorie des jeux et des jeux coopératifs combinée avec un algorithme K-moyennes. L'idée est que l'algorithme K-moyennes ne donne pas des résultats proches de l'optimal du fait de son initialisation effectuée au hasard. Cette initialisation est donc réalisée via un jeu coopératif dans lequel les joueurs sont les SOFs. L'initialisation via le jeu coopératif permet d'améliorer les résultats de K-moyennes. Cependant, cette valeur optimale peut être aussi obtenue en répétant l'exécution de l'algorithme K-moyennes un certain nombre de fois, ce qui limite l'intérêt de l'approche.

[TUNCER et collab., 2015] proposent pour illustrer un concept d'architecture de management et de contrôle distribuée et dynamique décrite en 2.2.5, une heuristique de placement de Local Controllers (LC) basée sur un critère de latence. L'algorithme est de type glouton et les LCs sont rajoutés au fur et à mesure en s'appuyant sur une métrique évaluant les meilleurs nœuds candidats. Le nombre maximum de contrôleurs à affecter est un paramètre d'entrée de l'algorithme. La condition de sortie utilisée réside dans le gain de latence intra-cluster obtenu par le rajout d'un contrôleur. Une variante de l'algorithme permet d'effectuer des re-configurations.

[SALLAHI et ST-HILAIRE, 2015] proposent un modèle MILP pour résoudre le problème de placement des contrôleurs sur des configurations très opérationnelles. Le modèle incorpore plusieurs types de contrôleurs (avec capacités de traitement et nombre de ports différents), et des informations de bande passante et de latence sur les différents liens connectant les SOFs aux contrôleurs. Le modèle incorpore aussi des coûts de déploiement et de raccordement des SOFs et des liens. L'objectif est de minimiser la somme des coûts en respectant des contraintes de latence et de bande passante. Le modèle ne permet de traiter que des problèmes de taille modeste jusqu'à 200 nœuds. Dans 10% des scénarios un temps de calcul d'une trentaine d'heures n'est pas suffisant.

Placement des contrôleurs sur un critère d'overhead du plan de contrôle [OBADIA et collab., 2015] construisent une heuristique gloutonne pour optimiser le placement des contrôleurs en cherchant à minimiser le trafic en overhead dans un réseau de contrôleurs distribués qui échangent nécessairement du trafic inband pour partager les états du réseau. Un modèle ILP permet de calibrer l'algorithme glouton. Le réseau qui connecte les contrôleurs en overlay est un spanning tree.

[ZHANG et collab., 2016] analysent de manière détaillée le rôle du protocole inter-

contrôleurs dans la latence induite pour descendre de nouvelles règles dans les SOFs. Deux types de protocoles sont possibles en fonction du niveau de cohérence des états du réseau que l'on souhaite assurer entre les contrôleurs ("strong" or "eventual"):

- Un protocole dans lequel un contrôleur unique est responsable de la mise à jour de la structure de données partagée et qui implique que les décisions de lecture/écriture remontent vers ce contrôleur.
- Un protocole dans lequel chaque contrôleur dispose d'une copie à jour de la structure de données. Dans ces conditions, des opérations de lecture/écriture ne nécessitent pas de remonter vers un contrôleur unique, mais un protocole de consensus est nécessaire.

Ces choix impliquent des réactivités différentes quand le contrôleur doit descendre des règles vers les SOFs. Un compromis est à trouver entre la réactivité et le niveau de cohérence des règles réseaux que l'on souhaite descendre vers les SOFs.

En fonction de la stratégie de consensus adoptée, le temps moyen d'activation d'une nouvelle règle dans un SOF peut atteindre plusieurs secondes sur un réseau à l'échelle d'un pays. Ceci justifie l'utilité d'un placement optimisé des contrôleurs dans le réseau pour réduire ce délai en cherchant à optimiser simultanément deux paramètres, la distance inter-contrôleurs et la distance SOFs à leur contrôleur.

[ZHANG et collab., 2016] implémentent aussi un modèle exact en s'appuyant sur un modèle ILP. Le modèle ILP permet de donner les fronts de Pareto pour les deux métriques avec de petites instances de réseaux. L'approche adoptée dans le modèle consiste à considérer les centroides contrôleurs et ensuite les SOFs qui leur sont attachés ce qui a l'avantage de réduire la combinatoire. C'est ce même modèle qui est ensuite adopté pour implémenter plusieurs variantes d'algorithmes évolutionnaires multi-objectifs. Le modèle adopté pour les solutions est alors une simple liste de nœuds contrôleurs.

Une première version de l'Algorithme Evolutionnaire (AE) se contente d'améliorer le front de Pareto au fil des itérations de l'algorithme en introduisant des variations au hasard sur le placement des contrôleurs par permutations. Il s'agit d'un échantillonnage au hasard de l'espace des solutions.

Dans une deuxième version, les perturbations sont itérées jusqu'à trouver une solution qui améliore le placement en réduisant le délai inter-contrôleurs. La solution peut ensuite être introduite pour mettre à jour le front de Pareto.

Une sous procédure permet de chercher une amélioration d'une solution courante en cherchant des contrôleurs qui réduisent la distance inter-contrôleurs. La deuxième version permet d'améliorer considérablement les performances de l'algorithme en introduisant un mécanisme simple d'optimisation locale.

Placement des contrôleurs sur critère de Latence & de charge La question de la charge des contrôleurs ou des clusters a été abordée par la suite par quelques auteurs.

Précisons les choses: si on compte le nombre de SOFs dans un cluster, on parle de taille de clusters et on a une variable entière dans le modèle; si on considère la charge de trafic de contrôle induite par chaque SOF, on a alors une variable réelle et on parle de charge des clusters. La charge générée par un SOF vers un contrôleur dépend ensuite du trafic. On fait généralement l'hypothèse que le trafic est homogène lorsqu'on considère des configurations statiques. Mais on peut considérer des approches dynamiques où le trafic varie au cours du temps.

[WANG et collab., 2018] proposent un algorithme K-moyennes pour minimiser la latence entre SOFs et contrôleurs.

Le papier aborde cependant la question de la charge des clusters indirectement en prenant en compte l'accroissement de la latence due à l'accroissement de la taille des files d'attentes dues au nombre de SOFs connectés aux contrôleurs.

L'équilibre de charge est en fait réalisé en partitionnant les clusters trop volumineux après avoir réalisé un premier placement, puis en prenant en compte l'accroissement de latence dû au remplissage des files d'attente du contrôleur.

Cette approche semble cependant limitée parce qu'elle fait l'hypothèse d'un trafic homogène généré par les différents SOFs vers les contrôleurs et aussi notamment parce qu'elle n'est illustrée que sur deux topologies.

[YAO et collab., 2014] introduisent le CCPP "Controllers Capacity Placement Problem". Il s'agit de minimiser la latence maximale en respectant une contrainte de capacité des clusters avec des SOFs qui induisent une charge variable. Pour cela ils réutilisent un algorithme itératif de résolution exacte du "vertex capacity K-center problem" proposé par [KHULLER et SUSSMANN, 1996] et [AYKUT OZSOY et PINAR, 2006] qui utilise une relaxation du modèle ILP.

Le principe de l'algorithme est de trouver une solution avec le nombre minimal de contrôleurs et ensuite de faire varier de manière itérative la taille des clusters en testant à chaque fois si la contrainte de capacité peut être tenue à partir de la résolution d'un sous modèle de type bin packing.

L'algorithme reste non polynomial mais permet de calculer des résultats exacts pour des instances de tailles raisonnables.

Dans cette approche, le nombre de "location facilities" (ce qui correspond à nos contrôleurs) est fixé préalablement et la capacité d'un cluster est traitée comme une contrainte. On ne doit pas attacher plus de q SOFs à un contrôleur. L'algorithme cherche ensuite à minimiser la latence.

L'approche proposée si elle permet d'atteindre une solution exacte en s'appuyant sur un solveur ILP comme CPLEX ne s'exécute pas en temps polynomial et nécessite sur certains problèmes de tailles moyennes de l'ordre d'une centaine de nœuds de nombreuses heures de calcul. Mais les auteurs proposent aussi d'adapter l'algorithme en discrétisant l'espace des distances entre SOFs et contrôleurs à des valeurs "faisables ou réalistes", ce qui permet de réduire fortement la combinatoire. L'approche est comparée à une approche dynamique présentée par [DIXIT et collab., 2013] et donne de meilleurs résultats.

[HU et collab., 2016], analysent de manière similaire l'impact d'une contrainte de charge sur le placement des contrôleurs en cherchant toujours à minimiser soit la latence moyenne soit la latence maximale.

Les auteurs réalisent cette fois une étude qualitative sur 10 topologies de taille moyenne en s'appuyant sur une modélisation MILP du problème et sur le solveur CPLEX. La contrainte de charge des contrôleurs est obtenue en divisant la charge globale par le nombre de contrôleurs.

Un indicateur de variance qui mesure l'écart par rapport à la valeur optimale permet d'évaluer l'équilibre de charge du réseau SDN.

La conclusion globale est que l'introduction d'une contrainte de charge n'impacte pas trop la latence avec une dégradation d'au plus 15% dans le pire des cas. Cette conclusion est cependant limitée aux 10 réseaux exemplaires choisis par les auteurs.

[KSENTINI et collab., 2016] abordent le problème du placement des contrôleurs en s'appuyant sur la théorie des jeux et le concept de jeu coopératif (bargaining game) où l'on cherche à échanger des marchandises de manière optimale en collaborant.

Trois objectifs sont recherchés, minimiser la latence entre SOFs et contrôleurs et l'overhead du plan de contrôle entre contrôleurs et inter-contrôleurs, et enfin garantir

dans le même temps un équilibre de charge.

La latence contrôleurs-SOFs et l'overhead inter-contrôleurs sont en fait deux objectifs contradictoires. Si l'on rapproche les contrôleurs entre eux et que l'on réduit leur nombre, on augmente nécessairement la latence entre contrôleurs et SOFs.

C'est ici que le jeu coopératif est introduit pour trouver le compromis entre les deux objectifs contradictoires précités pour placer les contrôleurs.

Il faut souligner qu'il ne s'agit pas d'une approche distribuée des contrôleurs mais simplement de la mise en œuvre d'une technique d'optimisation basée sur la théorie des jeux dans un contexte de deux objectifs contradictoires. Dans le modèle adopté, les deux joueurs qui coopèrent représentent la communication inter-contrôleur et la communication SOFs contrôleurs et non pas des SOFs ou des contrôleurs. L'évaluation de la technique proposée est basée sur un réseau simulé, dans lequel le trafic géré par chaque SOF est fixé aléatoirement. L'approche basée sur la théorie des jeux permet de réduire l'overhead de trafic global.

[YAO et collab., 2014] décrivent un exemple d'approche exacte MILP. En réalité dans ce travail, un solver MILP est utilisé comme sous routine dans le déroulement général d'un algorithme qui vise à optimiser à la fois la latence SOF-contrôleurs et la charge des contrôleurs.

[LANGE et collab., 2015b] et [HOCK et collab., 2013] ont proposé une approche qui constitue une sorte de référence pour la mise en œuvre d'une méta-heuristique qu'ils ont déclinée pour traiter différents scénarios de placement y compris dans une approche dynamique [HOCK et collab., 2014].

Ils proposent l'outil *POCO* qui permet de gérer un problème de placement de contrôleurs multi-objectifs en recherchant un compromis de placement des contrôleurs dans un WAN qui combine diverses métriques.

Les métriques considérées sont la latence contrôleurs - SOF, la latence inter-contrôleurs, l'équilibre ou le déséquilibre de charge. Ces métriques sont prises en compte de manière à optimiser la résilience du réseau en cas de pannes de contrôleurs ou de pannes de nœuds réseaux.

En effet, la métrique de déséquilibre de charge est considérée dans un cas normal et dans un cas de pannes ou une partie des nœuds réseaux peuvent être ré-alloués à des contrôleurs opérationnels. On a dans ce cas un problème quadri-objectifs.

POCO est implémenté avec Matlab et utilise une résolution MILP exacte pour les petites instances de problèmes et une méta-heuristique basée sur le simulated annealing multi-objectifs pour les instances de tailles plus élevées. Enfin l'approche dynamique présentée dans [HOCK et collab., 2014] montre la faisabilité de l'utilisation de l'outil pour des buts de ré-optimisation dans un réseau où la répartition des contrôleurs est basée sur le monitoring d'un réseau WAN à partir d'agents locaux remontant des informations de monitoring.

[JALILI et collab., 2015] se sont efforcés de décliner la même approche que POCO présentée en 2.3.1 avec les mêmes métriques à partir d'une méta-heuristique de type AEMO. Ils ont utilisé pour cela l'algorithme NSGA II de [DEB et collab., 2002a]. La représentation des individus adoptée ici est une simple liste de k contrôleurs. Le nombre de contrôleurs est fixé. Les SOFs attachés aux contrôleurs n'apparaissent pas dans la structure de données et sont déduits de sous routines qui ne sont pas explicitées. Ce travail présente de grosses limitations. Il n'est évalué que sur une seule instance de réseau de 34 nœuds et ne présente qu'une simple évolution du front de Pareto au cours des itérations de l'algorithme en le comparant avec une approche exacte MILP obtenue avec le solveur POCO de [LANGE et collab., 2015b].

[BO et collab., 2016] proposent une approche utilisant un algorithme génétique multi-objectifs pour résoudre le problème du placement des contrôleurs. Les deux métriques considérées sont d'une part, une métrique de charge qui intègre la charge induite dans les contrôleurs par chaque SOF du cluster, et d'autre part une métrique qui somme le poids du Minimum Spanning Tree de chaque cluster. Les deux métriques sont ensuite combinées linéairement par des coefficients pour exécuter en réalité un algorithme mono-objectif implémenté from scratch.

L'algorithme trouve un ensemble de k clusters. Le jeu sur les coefficients de la combinaison linéaire permet entre autre de densifier plus ou moins les clusters. Une deuxième routine trouve ensuite le centroïde de chaque cluster qui correspond au contrôleur. L'algorithme est testé sur 4 instances de réseaux et converge en quelques générations, avec parfois des paliers avant de trouver une nouvelle amélioration.

[YUAN et collab., 2018] abordent le problème du placement de contrôleurs sur un critère de latence et de charge. Le problème est formulé comme un problème de minimum weight matching dans un graphe bipartite, où d'un coté du graphe se trouvent les contrôleurs et de l'autre coté les SOFs. Le poids des arcs est la latence induite par ceux ci. Dans le modèle proposé la charge apparaît comme une borne supérieure d'un nombre de SOFs connectés au contrôleur.

Les auteurs proposent une approche exacte par énumération limitée à des problèmes de petites tailles et utilisant un algorithme exact pour calculer le minimum matching. Pour les problèmes de plus grande taille, ils utilisent un AE qui utilise comme sous routine l'algorithme exact précité. Les opérateurs de variations de l'AE consistent à partager les contrôleurs des parents ou à remplacer un contrôleur par un autre dans la liste.

Placement des contrôleurs sous critères de latence, charge & fiabilité Le fait de contrôler des sous parties du réseau à l'aide de contrôleurs centralisés modifie l'approche de la fiabilité et nécessite en particulier de définir les métriques adaptées pour garantir la fiabilité de l'infrastructure SDN globale.

[GUAN et collab., 2013] analysent les problématiques de fiabilité et de scalabilité dans le contexte du réseau GENI ([BERMAN et collab., 2014]) qui est une architecture SDN à la fois distribuée et hiérarchisée. Ils mettent en œuvre une expérimentation qui montre que l'arrivée d'un flux trop important de paquets inconnus non routables directement par les SOFs peut provoquer un effondrement du réseau et des pertes de paquets importantes.

A partir de 250 paquets nouveaux par secondes, les pertes atteignent déjà 20%. Ce travail justifie l'importance à accorder aux problématiques de fiabilité et de scalabilité: capacité de traitement des contrôleurs et des switches, goulots d'étranglement du plan de contrôle SDN, dimensionnement des clusters de contrôleurs, services traités.

[HU et collab., 2013] adoptent une approche probabiliste pour la fiabilité et définissent la métrique: "expected percentage of control path loss", qui correspond au pourcentage de chemins de contrôle devenus défectueux du fait d'une perte de liens ou de nœuds.

Les hypothèses adoptées sont les suivantes:

- Le cas où plusieurs liens ou nœuds tombent simultanément n'est pas envisagé,
- Les évènements de la couche physique provoquant les pannes sont statistiquement indépendants,
- Un élément physique défectueux fait tomber le control path qui le traverse, et il n'y a pas de mécanismes de secours ou de récupération.

Le papier compare ensuite différents algorithmes sur un ensemble de topologies WAN: un algorithme qui tire les placements aléatoirement, un algorithme glouton, une meta-heuristique basée sur le simulated annealing et finalement une approche en force brute.

L'approche basée sur une méta-heuristique donne les meilleurs résultats.

L'autre conclusion intéressante est que trop peu de contrôleurs mis en œuvre dégradent les performances car l'impact d'une panne augmente mais il existe aussi une tendance inverse quand le nombre de contrôleurs devient trop élevé, car le nombre de chemins de contrôle est alors impacté ce qui augmente aussi la probabilité de panne. Les algorithmes présentés ne prennent pas en compte l'équilibre de charge entre les contrôleurs. Le seul critère est l'attachement du SOF au contrôleur le plus proche, au sens du plus court chemin.

[ROS et RUIZ, 2014] abordent de manière frontale le problème d'assurer une fiabilité de niveau élevée de 99,999% dans un réseau SDN avec une architecture distribuée.

Le problème est intitulé: "Fault Tolerant Controller Placement problem". Dans un contexte où un SOF peut être rattaché à plusieurs contrôleurs pour assurer une fiabilité suffisante, la question posée est combien de contrôleurs faut-il et où les placer pour une topologie donnée, tout en minimisant leur nombre. Il faut donc qu'un chemin de contrôle parmi tous ceux possibles continue d'exister en cas de pannes pour chaque SOF du réseau avec une probabilité suffisante.

La fiabilité est définie ici sur la base du concept de "k terminal reliability" [AYOUB et collab., 2000] qui se définit comme la probabilité que toutes paires de terminaux dans un ensemble de k terminaux puissent communiquer entre elles.

A chaque nœud et lien du réseau est attaché une probabilité de panne. Aux nœuds susceptibles d'accueillir un contrôleur est attaché un coût de déploiement d'un contrôleur.

Le problème est NP difficile et sa formulation pour la contrainte de fiabilité est non linéaire, puisqu'il s'agit d'un produit de probabilités.

Les auteurs proposent une heuristique pour résoudre le problème garantissant que la contrainte de fiabilité est vérifiée et en classifiant les solutions qui minimisent le nombre de contrôleurs. Une représentation équivalente du graphe permet de calculer en temps polynomial une borne inférieure de la "k terminal reliability" pour chaque nœud connecté à un contrôleur. L'heuristique permet ensuite de calculer un ensemble de solutions qui vérifient cette borne et ces solutions sont ensuite classifiées pour minimiser le coût de déploiement.

L'évaluation est conduite sur des topologies extraites de la base de données zoo topology [KNIGHT et collab., 2011] qui sont connexes. Chaque nœud peut supporter plusieurs contrôleurs et dans l'évaluation, cette limite est fixée à deux. Le nombre de contrôleurs est fortement dépendant de la topologie qui détermine le nombre de chemins redondants possible. Contrairement à la conclusion des auteurs, ils nous semble que le nombre de contrôleurs à mettre en oeuvre pour assurer cette "five nine reliability" est plutôt élevé (jusqu'à un contrôleur pour 3 SOFs pour certaines topologies) ce qui enlèverait tout intérêt à déployer des contrôleurs pour réaliser un réseau SDN et militerait pour s'en tenir à une approche classique des réseaux distribués. De plus il s'agit d'une heuristique qui n'optimise que la seule métrique de fiabilité.

[JIMÉNEZ et collab., 2014], cherchent à combiner dans leur étude à la fois une métrique de latence et une notion de robustesse du cluster auquel est associé chaque contrôleur.

Ils proposent une heuristique K-critical qui tend aussi à minimiser le nombre de contrôleurs.

Pour garantir la robustesse l'idée est que le spanning-tree qui distribue le flux de con-

trôle entre le contrôleur et les nœuds soit à la fois large avec des branches courtes et équilibré, ce qui doit limiter les nœuds impactés en cas de défection d'un nœud ou d'un lien. Pour cela une fonction est insérée dans les itérations qui contrôle la longueur des branches du spanning tree.

Si l'heuristique de ce papier est intéressante car elle permet de construire des clusters résistants aux pannes, elle ne garantit pas cependant que le nombre de contrôleurs soit au minimum.

[LIAO et collab., 2017] proposent d'utiliser une approche de Density based clustering pour définir le placement des contrôleurs dans un réseau.

Leur proposition combine en réalité plusieurs algorithmes et prend en compte les métriques suivantes : la latence contrôleur-SOFs, la latence inter-contrôleurs, la charge des contrôleurs qui doit rester bornée par rapport à la capacité du contrôleur, et le risque de déconnexion d'un SOF en cas de pannes sur un ou deux liens. L'idée est aussi que plus le cluster est dense moins la probabilité de panne est élevée.

La première étape consiste à trouver le nombre de clusters à partir d'un paramètre de densité déterminé expérimentalement sur la base de données zoo-topologie. Une fois le nombre k de clusters déterminé, les clusters sont créés par proximité avec les nœuds dont la densité est la plus élevée. Le critère de charge est introduit en ajustant l'affectation des SOFs aux contrôleurs pour respecter la contrainte de charge. Enfin comme la position des contrôleurs peut être ajustée à l'intérieur des clusters on peut optimiser la latence inter-contrôleurs.

Le gain de l'approche en terme de taux de pannes est établi expérimentalement et reflète que les clusters ont une densité plus élevée que la moyenne du graphe. Les auteurs établissent des comparaisons avec d'autres approches et leur proposition apparaît comme performante, le temps de calcul en particulier est relativement raisonnable de l'ordre de $O(n^2)$. Certaines courbes qui donnent un front de Pareto sont discutables (average et worse case latency), en effet si le front de Pareto est mieux situé les points couvrent en réalité des valeurs moins intéressantes, et surtout les courbes ne concernent que deux réseaux.

[KILLI et RAO, 2017] proposent de minimiser la somme des latences au premier contrôleur de référence (disposant d'une capacité suffisante) et des latences au second contrôleur disponible en terme de capacité s'il y avait défection du premier contrôleur.

L'objectif est de minimiser la latence la plus mauvaise (worse case latency) en cas de défection d'un contrôleur. L'approche est étendue au cas de pannes multiples sur des contrôleurs. Un modèle MILP est proposé ainsi qu'une méta-heuristique de résolution basée sur le "simulated annealing". Une comparaison du modèle MILP avec la méta-heuristique est effectuée sur une instance de réseau de taille moyenne qui donne de bons résultats. La méta-heuristique est aussi plus rapide que la résolution exacte.

[LANGE et collab., 2015a] proposent une heuristique susceptible de traiter un problème multi-objectifs de placement de contrôleurs dans un temps raisonnable.

L'approche prend en compte les deux objectifs classiquement considérés qui sont, la latence moyenne contrôleurs-SOF, et l'équilibre de charge. Le nombre de contrôleurs est cependant fixé. Les auteurs proposent l'algorithme Capacited K-medoids qui cherche à optimiser de manière itérative, la latence SOF-contrôleur, et la charge des clusters.

Ils modifient pour cela l'algorithme K-medoid que nous décrivons en 4.2.3. L'algorithme prend en paramètre d'entrée le nombre k de contrôleurs et une tolérance admissible de différence de capacité entre clusters. Les itérations successives visent à réduire le déséquilibre de capacité en introduisant celui-ci comme une contrainte dans l'exécution de l'algorithme. L'heuristique proposée est évaluée en testant différentes

bornes de déséquilibres de charge pour trouver un front de Pareto. Le front de Pareto est ensuite comparé avec une méta-heuristique basée sur le simulated annealing décrite par [LANGE et collab., 2015b].

[XIAO et collab., 2014] abordent le problème du placement avec une technique sophistiquée de "spectral clustering". Il s'agit là encore d'une technique de clustering adaptée pour résoudre le problème du placement des contrôleurs. Le nombre de contrôleurs est fixé.

La technique de clustering spectral permet de partitionner des données en les projetant dans un espace vectoriel de dimension moins élevée construit à partir des k vecteurs propres de la matrice Laplacienne tirée de la matrice de similarités.

Une fois projeté dans cet espace de dimension k, l'algorithme applique un algorithme K-moyennes sur les objets représentés par les lignes de cette matrice pour construire des clusters. Les auteurs utilisent ici comme matrice de similarités du graphe la matrice de latence entre les nœuds du graphe et une pondération attachée à chaque lien qui représente la bande passante.

L'algorithme cherche à maximiser la similarité à l'intérieur des k clusters en maximisant la bande passante et en la minimisant entre les clusters.

L'algorithme semble équilibrer efficacement les clusters en maximisant la densité (edge density) dans les différents clusters.

Comme souvent, le nombre de clusters k est fixé en entrée de l'algorithme. Le critère de latence sert ensuite aussi à trouver la position du contrôleur de chaque cluster. Il faut remarquer que l'algorithme ne cherche pas explicitement à faire en sorte que la latence contrôleurs – SOF soit minimisée mais plutôt à équilibrer les clusters sur un critère de densité intra et inter-clusters.

Les auteurs considèrent que maximiser la edge density intra-cluster augmente la fiabilité, ce qui peut-être discuté dans la mesure où ce n'est pas explicitement une métrique de connectivité.

[RATH et collab., 2014] proposent une approche basée sur la théorie des jeux et le principe d'un jeu à somme nulle pour proposer un algorithme distribué sur chaque entité contrôleur.

L'idée est que chaque contrôleur se compare à ses voisins pour prendre des décisions d'activation ou de désactivation des contrôleurs et de ré-allocation ou de redéploiement, des SOFs.

La comparaison s'effectue sur la base d'une fonction de coût qui évolue en suivant l'évolution du trafic et qui reflète une métrique de latence et une métrique de charge attachés aux contrôleurs. L'idée est de minimiser la latence tout en maximisant la charge en répartissant celle-ci entre un contrôleur et ses voisins.

Le mécanisme permet d'optimiser la latence entre contrôleurs et SOFs et la charge des contrôleurs dans un scénario dynamique.

L'initialisation du premier placement des contrôleurs est réalisée de manière centralisée. Ensuite chaque contrôleur optimise au mieux en dialoguant avec ses voisins les deux métriques considérées et en jouant un jeu à somme nulle avec ceux-ci. Il faut remarquer qu'il s'agit plutôt d'un scénario de type datacenter avec des contrôleurs virtualisés.

Prise en compte de la résilience [GUO et BHATTACHARYA, 2013] introduisent pour traiter de la résilience des réseaux SDN, le concept de réseaux de contrôle (CS pour controllers-switches network) et de réseaux de données (SS pour switches-switches network) interdépendants.

Un graphe d'interdépendance est construit qui relie les nœuds des deux réseaux et qui met en évidence l'interdépendance de CS et SS du point de vue des pannes qui peuvent se produire et dérouler des effets en cascades.

La résilience est introduite comme une métrique qui mesure le nombre de nœuds qui restent opérationnels dans l'état stable qui est atteint après une succession de défections suite à une panne d'un nœud ou d'un lien.

La résilience mesurée est la résilience moyenne obtenue sur les différents clusters après placement des contrôleurs en prenant aussi en compte la résilience globale du réseau.

L'algorithme proposé est une heuristique gloutonne avec en entrée le nombre de contrôleurs.

[ZHANG et collab., 2011] analysent expérimentalement l'impact du placement des contrôleurs sur la résilience ou la robustesse d'un réseau Software Defined WAN (SD-WAN) .

Une des hypothèses du travail est qu'un SOF n'est plus fonctionnel s'il n'est plus connecté à son contrôleur.

L'analyse vise à évaluer le nombre de nœuds déconnectés en cas de panne d'un nœud dans le cas d'une architecture réseau traditionnelle et dans le cas d'une architecture SDN.

La résilience est évaluée par rapport au nombre de chemins disjoints à l'intérieur d'un domaine. Les auteurs proposent ensuite un algorithme min-cut modifié qui permet de maximiser la résilience.

Cet algorithme est en fait un algorithme de clustering qui partitionne le réseau en clusters avec une connectivité élevée tandis que la connectivité inter-clusters est faible. Le nombre de contrôleurs est fixé.

Dynamicité du placement de contrôleurs Il n'est pas absurde d'envisager un scénario de placement dynamique des contrôleurs soit du fait de modification de la topologie du réseau due à des pannes, soit pour des raisons de re-configuration du fait des modifications des caractéristiques du trafic au cours du temps qui induisent une évolution de la charge du plan de contrôle.

On peut vouloir modifier la topologie des contrôleurs, rajouter ou diminuer le nombre de contrôleurs pour réduire la consommation ou répondre à un risque de congestion ponctuel.

[DIXIT et collab., 2014] envisagent une approche dans laquelle les clusters de SOFs autour des contrôleurs peuvent être reconfigurés de manière dynamique en fonction de la charge, en s'appuyant sur les rôles maître/esclave que peuvent adopter les contrôleurs OF.

Les auteurs proposent un protocole et une architecture mais aussi un ensemble d'algorithmes permettant de diminuer ou d'accroître leur nombre en les activant ou en les désactivant mais aussi de réaffecter les SOFs autour des contrôleurs pour améliorer l'équilibre de charge.

L'algorithme de rééquilibrage de charge ne cherche pas à résoudre un problème d'optimisation mais plutôt à maintenir la charge entre des seuils hauts et bas fixés permettant d'assurer une capacité de traitement.

Bien que très abouti, l'approche proposée ne donne pas cependant de stratégie permettant de placer les contrôleurs de manière optimale et indique simplement que lorsqu'un nouveau contrôleur est activé il doit être localisé à proximité du contrôleur trop chargé pour éviter d'introduire des problèmes de latences trop importantes.

[YAO et collab., 2015] proposent une métrique permettant d'assurer la migration de SOFs vers un autre contrôleur en cas de variation de charge. La métrique reflète le coût induit par la génération de message Packets-in par chaque SOF. Elle prend en compte le degré du nœud SOF et le coût de routage entre le SOF et le nœud contrôleur.

Pour déployer des contrôleurs distribués, le graphe est partitionné en estimant le nombre de contrôleurs en utilisant une technique de partitionnement de graphe exploitant la métrique (multi-level k-way partitionning).

Enfin un algorithme de migration de SOFs est proposé qui permet de modifier la répartition des SOFs et qui prend en compte la capacité des contrôleurs. Un module de monitoring attaché à chaque contrôleur permet d'évaluer la charge induite par les SOFs.

Conclusion sur le placement des contrôleurs [WANG et collab., 2017] proposent un survey du sujet assez complet avec une tentative de classification des différentes techniques proposées.

En conclusion, on peut constater le large éventail de techniques mises en œuvre pour traiter du problème de placement de contrôleurs.

Ce n'est pas surprenant puisqu'on est généralement confronté à un problème NP-difficile quel que soit les déclinaisons abordées.

Ces techniques vont d'heuristiques diverses souvent inspirées des techniques de clustering appliquées à un modèle de graphe, en passant par des techniques utilisant des solveurs et des modèles de programmation linéaire en nombres entiers et enfin à des méta-heuristiques qui ont l'avantage d'être plus génériques et de pouvoir traiter des problèmes multi-objectifs et multi-contraints.

Les approches exactes à base de programmation linéaire sont généralement limitées par le problème du temps de calcul mais des approches mettant en œuvre du parallélisme sur des cartes GPU par exemple ne sont pas du tout explorées.

Il faut aussi remarquer que la programmation linéaire peut aussi être utilisée comme une sous routine d'une heuristique de résolution.

La multiplicité des problèmes traités notamment en terme de métriques (latences diverses, résilience ou fiabilité, charge, overhead de trafic de contrôle) rend difficile de tirer une règle générale de tous ces travaux.

Cependant on voit bien que la mise en œuvre d'une méta-heuristique performante comme les techniques stochastiques, permet d'aborder des problèmes plus complexes et multi-objectifs de manière assez générique même si on n'a pas l'assurance d'obtenir le résultat exact. L'exemple POCO présenté en 2.3.1 qui cherche à optimiser trois objectifs importants est le plus abouti.

Quand il s'agit de placer plusieurs contrôleurs de manière distribuée dans un réseau, l'idée vient naturellement d'envisager une approche distribuée comme on le fait pour le problème du routage dans les réseaux classiques. Cependant, il n'y a pas de propositions satisfaisantes. Il faut noter cependant l'approche 2.3.1 qui utilise la théorie des jeux pour équilibrer dynamiquement le trafic en réaffectant des SOFs ou en activant désactivant des contrôleurs.

Il manque aussi une approche suffisamment générique du problème de placement des contrôleurs permettant de traiter à la fois les différentes métriques avec des aspects multi-objectifs et des aspects dynamiques.

.

2.3.2 Placement de chemins

Nous entendons par placement de chemins le problème d'établissement de circuits virtuels dans un réseau pour écouler différent types de trafic. Cependant, nous nous restreignons ici au problème du routage avec qualité de service. Il s'agit dans ce cas de placer des chemins dans un réseau en respectant un ensemble de contraintes et en cherchant à optimiser un ensemble d'objectifs de QoS.

Le PCE, un outil SDN d'ingénierie de trafic

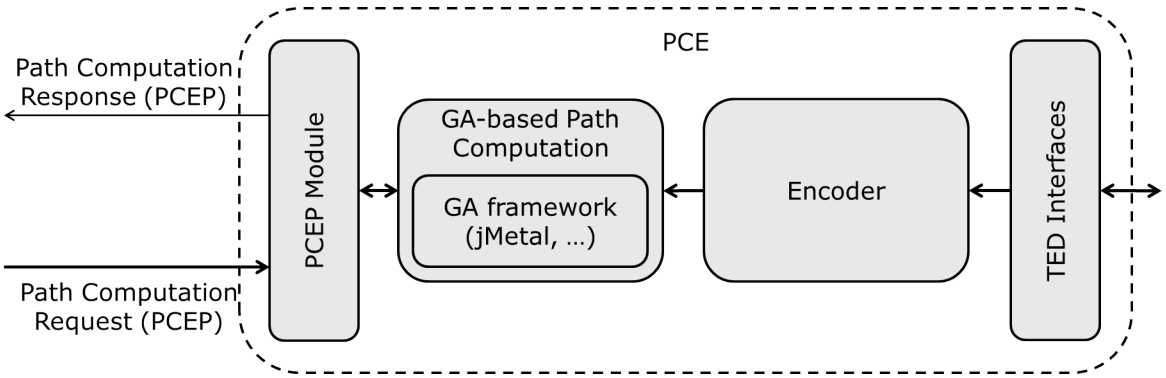


Figure 2.7 – Path Computation Element avec un Algorithme évolutionnaire

Nous présentons succinctement ici le Path Computation Element (PCE) décrit par [FARREL et collab., 2006], c'est un outil d'ingénierie de trafic centralisé pour les réseaux Multiprotocol Label Switching (MPLS). Cet aspect centralisé permet d'exercer un contrôle global sur le trafic via une vue globale du réseau, qui en fait un outil que l'on peut considérer comme SDN compatible.

Le PCE communique avec ses routeurs clients qui sont les intermédiaires chargés du déploiement des chemins MPLS (LSP) pour le trafic dans le réseau. Il utilise pour cela le protocole Path Computation Element Protocol (PCEP). Ce concept peut être implémenté au dessus d'un contrôleur SDN ou embarqué, comme dans le cas du contrôleur OpendayLight.

Le PCE s'appuie sur les données de topologies fournies par une Traffic Engineering Database (TED). L'outil peut héberger potentiellement des algorithmes d'ingénierie de trafic comme les algorithmes que nous discutons ici. Des variantes d'implémentation existent suivant que le PCE connaît ou ne connaît pas l'état courant d'établissement des LSPs dans le réseau. S'il le connaît il gère dans ce cas une LSP database. Un schéma illustratif est présenté en 2.7.

Problème traité et approches proposées

Les objectifs à optimiser considérés sont ici des métriques additives attachées aux chemins. Il s'agit d'une approche restreinte par rapport à une approche plus générale dans laquelle on prendrait en compte des métriques globales à l'ensemble du réseau comme par exemple la bande passante résiduelle.

Plusieurs approches ont été proposées dans la littérature pour résoudre le problème du routage avec QoS. On peut les classer en deux groupes principaux : les approches avec heuristiques et les approches avec un algorithme exact.

[LEE et collab., 1995] introduit l'algorithme Fallback (FB). Il calcule le chemin le moins coûteux pour chaque métrique et vérifie ensuite si ce chemin est réalisable. L'algorithme s'arrête lorsque le chemin le plus court satisfait toutes les contraintes. Sinon, il poursuit avec une autre métrique de QoS jusqu'à ce qu'un chemin réalisable soit identifié ou que toutes les mesures de QoS soient examinées et respectent les contraintes.

La complexité de l'algorithme dans le pire des cas est M fois celle de Dijkstra, où M est le nombre de mesures de QoS. Cette approche fonctionne bien lorsque les différentes métriques des liens sont positivement corrélées [KUIPERS et collab., 2002].

[KORKMAZ et KRUNZ, 2001] proposent une autre heuristique pour trouver un chemin optimal multi QoS, *H_MCOP*. Cet algorithme tente de trouver un chemin réalisable pour un nombre quelconque de contraintes tout en minimisant simultanément une fonction non linéaire des longueurs du chemin relativement aux différentes métriques. La recherche d'un parcours réalisable se fait par approximations successives.

H_MCOP exécute deux versions modifiées de l'algorithme de Dijkstra, une dans le sens direct et une en sens inverse en calculant le chemin le plus court de chaque nœud à la destination pour une combinaison linéaire des différentes métriques.

[NEVE et MIEGHEM, 2000] proposent un algorithme de routage à contraintes multiples et de précision ajustable (TAMCRA).

Il repose sur trois concepts fondamentaux : une mesure non linéaire de la longueur du chemin, les k shortest paths, et le principe de retenir des chemins non dominés. La mesure non linéaire de la longueur du chemin est efficace lorsque les métriques du chemin ne sont pas corrélées [KUIPERS et collab., 2002].

TAMCRA conserve les k chemins non dominés pour chaque nœud intermédiaire i sur le chemin de la source à la destination, pour les réutiliser si besoin.

[MIEGHEM et collab., 2001] proposent Self Adaptative MultiOjective Constraint Routing Algorithm (SAMCRA) qui est une variante exacte de TAMCRA. SAMCRA diffère de TAMCRA en adaptant le nombre de chemins stockés à chaque nœud. Comme algorithme exact, SAMCRA garantit qu'un chemin réalisable et optimal est trouvé s'il existe.

[LIU et RAMAKRISHNAN, 2001] introduisent A*prune autre algorithme exact.

Pour chaque mesure de QoS, A*prune calcule les chemins les plus courts entre la source et tous les autres nœuds du réseau.

Les métriques de ces chemins sont exploitées pour évaluer si un sous chemin donné peut devenir un chemin réalisable.

Le nœud pour lequel le chemin pour l'atteindre est le plus court est retiré d'une pile puis tous ses voisins sont vérifiés. Les voisins qui introduisent une boucle ou une violation des contraintes sont supprimés. L'algorithme continue jusqu'à ce que le chemin le plus court avec les K contraintes soit trouvé ou que la pile soit vide.

[KUIPERS et collab., 2004] comparent les performances des différents algorithmes mentionnés.

Pré-requis sur SAMCRA Certains des concepts de l'heuristique SAMCRA présentés par [VAN MIEGHEM et KUIPERS, 2004] ont été utilisés dans la contribution présentée en 4.5.1. C'est pourquoi nous avons pensé utile de présenter certaines des notions mises en œoeuvre dans cette heuristique.

La structure de référence reste l'algorithme polynomial de Djikstra de recherche d'un plus court chemin enrichi d'un certain nombre de concepts permettant de traiter l'aspect multi-contraint et multi-objectifs. Ils sont présentés ci-dessous.

Définition non linéaire de la longueur du chemin On s'appuie pour l'algorithme sur une définition non linéaire de la notion de longueur d'un chemin combinant les différentes métriques.

Pour cela, on pose $\vec{c}$ un vecteur de dimension m dont chaque coordonnée est un poids associé à chacune des m métriques que l'on souhaite prendre en compte et on considère un chemin donné p.

On note L_i les différentes contraintes, avec $1 \leq i \leq m$.

On note $\vec{d}$ le vecteur qui représente les différentes métriques du chemin.

La longueur d'un chemin peut alors s'exprimer comme le produit scalaire de d et c: $l(p) = \vec{c} \cdot \vec{d}$.

Il s'agit de raisonner comme si l'on disposait d'une seule métrique réalisée comme une combinaison linéaire des différentes métriques et appliquer un algorithme de Djikstra.

L'inconvénient de cette approche est qu'elle ne permet pas nécessairement de trouver une solution optimale car il s'agit de rechercher un front de Pareto et qu'elle peut trouver des solutions qui ne respectent pas les contraintes comme indiqué par [VAN MIEGHEM et KUIPERS, 2004].

Pour cette raison on fait appel à la norme infinie comme fonction non linéaire normalisée par rapport aux contraintes:
$l_\infty(p) = \max_{1 \leq i \leq m} \frac{c_i(P)}{L_i}$.

Cette norme garantit que la solution trouvée respectera les contraintes.

K-shortest path La notion de K-shortest Path est présentée par [EPPSTEIN, 1999]. L'idée est qu'au lieu de stocker le plus court chemin dans chaque nœud intermédiaire pour atteindre un nœud donné quand on applique Djikstra, on stocke les k plus courts chemin en commençant par le plus court. k doit bien sûr être borné car sinon le nombre de chemins possible peut être très grand.

Dominated path La notion de dominance pour un chemin. Lorsqu'on veut comparer deux chemins P1 et P2 vers un nœud intermédiaire, on fait appel au concept de non dominance et l'on dit que P1 domine P2 si et seulement si $\forall i\, w_i(P1) \leq w_i(P2)$. D'autre part on dira qu'un chemin P est non dominant s'il n'existe pas un chemin P_j avec $\forall i,\ c_i(P_j) \leq c_i(P_j)$.

On remarquera enfin que si l'on considère la norme infinie, alors si P2 est dominé par P1, P2 ne pourra pas être un sous chemin d'un chemin optimal depuis le nœud source jusqu'au nœud destination.

Look ahead L'idée ici est d'exécuter un algorithme de Djikstra depuis le nœud destination vers chaque nœud du graphe pour chacune des m métriques et de stocker le vecteur résultat dans chaque nœud.

Ce principe permet de supprimer des nœuds candidats dans le chemin en vérifiant rapidement que le chemin global ne vérifierait pas dans ce cas les contraintes spécifiées.

Ce principe permet pour des tailles de problèmes importantes d'accélérer les traitements.

Complexité Nous ne détaillons pas l'exécution de l'algorithme SAMCRA qui est décrit en détail par [VAN MIEGHEM et KUIPERS, 2004]. La complexité dans le pire cas, s'exprime de la manière suivante, avec E le nombre d'arcs du graphe, N, le nombre de nœuds du graphe et m, le nombre de métriques.

On a dans ce cas:

$$C_{samcra} = O(k_{max} \cdot N \cdot log(kN) + k_{max}^2 \cdot mE) \tag{2.1}$$

Conclusion sur le placement de chemins

Nous avons abordé ici le problème du routage avec qualité de service et les heuristiques exactes développées pour cela.

Une approche plus large du sujet aurait nécessité de regarder le problème du placement de chemins dans son cas le plus général.

L'idée est dans ce cas de trouver des méthodes pour optimiser des ressources globales (bande passante ou autre) dans un réseau lorsqu'on y déploie des réseaux virtuels. On peut le formuler aussi comme un problème de définition de circuits virtuels ou VPN dans un réseau. Un exemple d'approche par programmation linéaire en nombres entiers (MILP) de ce problème avec un un mécanisme de relaxation a été proposé par [ZHU et AMMAR, 2006].

2.3.3 Placement de chaînes de VNF

L'idée n'est pas de traiter le sujet de manière exhaustive mais de présenter la problématique et quelques exemples de manières de l'aborder.

Dans le modèle adopté ETSI-NFV, issu de l'ETSI, un service réseau est modélisé comme un graphe assemblant différentes fonctions réseaux virtualisées VNF, appelé VNF-FG.

Le problème du placement du service réseau revient alors en fait à mapper le VNF-FG sur un modèle de graphe du réseau qui intègre les ressources disponibles, en prenant en compte différentes contraintes et objectifs propres au service réseau.

Les objectifs peuvent être soit globaux comme par exemple la question de la répartition du trafic dans le réseau ou la consommation énergétique, soit locaux comme les métriques de QoS fixées pour le service que l'on cherche à déployer.

Modèle ETSI NFV MANO

Notre référence est le modèle ETSI NFV, Management And Network Orchestration (MANO) [ETSI, 2014] comme contexte de mise en œuvre pratique des algorithmes de placement de chaînes de VNF proposés. MANO est composé de quatre types d'éléments :

- le gestionnaire d'infrastructure virtualisée VIM,
- le VNF Manager (VNFM),
- l'orchestrateur, NFV Orchestrator (NFVO),
- le gestionnaire de WAN, WAN Infrastructure Manager (WIM).

L'architecture peut intégrer plusieurs VIM ou VNFM, mais un seul NFVO.

Un VIM est responsable de la gestion des ressources d'une infrastructure NFV Infrastructure (NFVI). Un NFVI est, typiquement, composé d'un datacenter, ou d'un ensemble de datacenters inter-connectés. Afin d'interconnecter les différents NFVI gérées par les VIMs, il est nécessaire d'introduire un autre gestionnaire de réseaux WAN, le WLAN Infrastructure Manager WIM. Ce dernier peut être considéré comme un VIM qui ne gère que des ressources réseau.

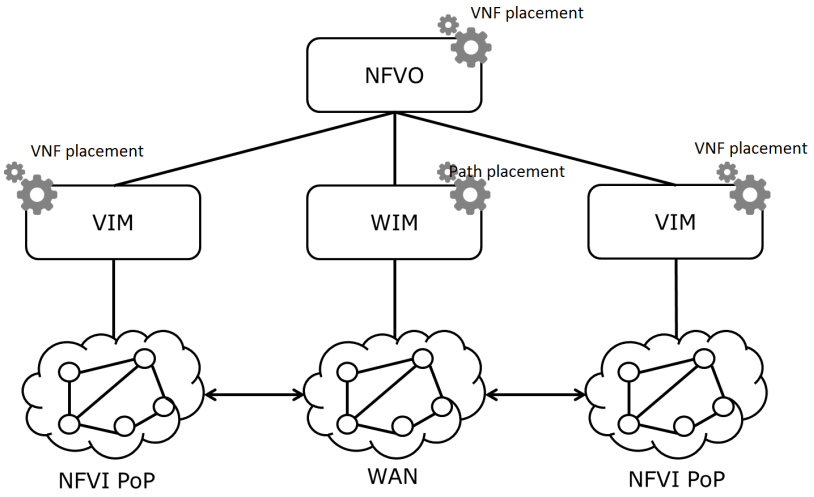


Figure 2.8 – NFV MANO framework

Les VIM et les WIM connaissent les ressources dont ils sont responsables et exposent une abstraction de celles-ci à leurs interfaces nord, auxquelles le VNFM et le NFVO ont accès. La gestion des ressources comprend la découverte de celles-ci, leur allocation, leur libération, les activités de monitoring et de gestion des fautes.

Les VNFM sont responsables de la gestion du cycle de vie des VNFs. Les VNFMs ne sollicitent pas directement des ressources, celles-ci sont gérées par le NFVO. Par conséquent, ils ne sont pas responsables du placement des VNFs dans l'infrastructure.

Déploiement d'un service réseau

Le NFVO est le point d'entrée de MANO pour les utilisateurs. Il reçoit les demandes de services réseau et convertit chacune d'entre elles en un ensemble de VNF interconnectées appelé VNF-FGs, autrement dit, des chaînes de VNFs.

En se basant sur l'abstraction des topologies fournies par les VIM et les WIM, il choisit ensuite où ces VNFs doivent être placées et aussi comment elles doivent être reliées.

Une fois ce placement décidé, le NFVO réserve les ressources correspondantes des VIM et les fournit aux VNFM. Le déploiement effectif des VNFs est effectué par les VNFM. Le NFVO est alors responsable de la gestion du cycle de vie de la VNF-FG et met à disposition des informations de monitoring à son interface nord.

Le NFVO est responsable du déploiement du graphe de VNFs sur les différents VIMs, ce qui est le travail typique d'un algorithme de placement de VNF-FG.

De plus, un tel algorithme peut également être utilisé par les VIM. En effet, les VIM reçoivent des demandes de ressources qui doivent être réparties dans la NFVI, ce qui est très similaire au problème d'allocation des VNF-FG. En effet, placer une fonction, c'est lui allouer des ressources de la NFVI. En ce qui concerne les WIMs, ils peuvent utiliser des algorithmes de calcul de routes ou de placement de chemins comme décrit dans le passage précédent. Le schéma d'architecture 2.8 illustre toutes ces idées.

Exemple du problème de placement de machines virtuelles

Le problème du placement de machines virtuelles ou de fonctions logicielles dans une infrastructure peut être vu comme un sous problème du problème de placement de VNF-FG au même titre que le problème de placement de chemins. Nous présentons ici un

seul exemple très parlant qui met en œuvre une stratégie à base de méta-heuristiques évolutionnaire.

[JAYASENA et collab., 2018] proposent une approche à base de méta-heuristiques multi-objectifs pour placer de manière optimale des machines virtuelles en prenant en compte trois objectifs: un objectif d'optimisation de la consommation énergétique globale, un objectif de réduction du nombre de migrations de machines virtuelles en cas de surcharge, et enfin un objectif visant à optimiser la QoS des services liés au déploiement des machines virtuelles.

Les auteurs proposent une méta-heuristique basée sur le concept de "Virus colony search" qui simule la diffusion et l'évolution d'une infection dans une population de cellules.

Il s'agit d'une méta-heuristique évolutionnaire basée sur l'initialisation d'une population comme les AE standards. La méta-heuristique est ensuite comparée à deux AE plus classiques implémentés pour atteindre le même résultat, NSGA-II d'une part et d'autre part, un algorithme MOEAD basé sur la décomposition en combinaisons linéaires des objectifs recherchés.

L'objectif de consommation énergétique est exprimé comme une relation linéaire qui prend en compte le nombre de CPUs mis en œuvre dans une machine physique et la consommation des VMs.

L'objectif de QoS est calculé par rapport à un taux de surcharge des machines physiques et est combiné par produit avec le nombre de migrations pour constituer un seul objectif à minimiser. On a donc finalement trois objectifs à optimiser quand on inclus la minimisation du nombre de VMs.

L'algorithme proposé est structuré en trois parties: un mécanisme de diffusion aléatoire basée sur une marche aléatoire gaussienne pour explorer l'espace de recherche, un mécanisme Covarariance Matrix Adaptation Evolution Strategy (CM-ES) comme décrit par [AUGER et HANSEN, 2005], et enfin un mécanisme simulant la réponse immunitaire d'un organisme pour éliminer les moins bonnes solutions.

L'évaluation s'appuie sur différents indicateurs de qualité dont l'hyper-volume et est basée sur l'outil CloudSim qui fournit des ensembles de données d'évaluation de références. Elle montre de meilleurs résultats pour l'algorithme proposé que pour les algorithmes AEMO, NSGA-II et MOEAD ([ZHANG et LI, 2007]).

Exemples de traitement du problème du placement de VNF-FGs

En préalable on peut remarquer que le problème de placement de VNF-FGs combine les deux aspects déjà évoqués: un problème de location facilities ou de bin packing (sur quels nœuds placer les VNFs en respectant certaines contraintes) [MANN, 2015] et un problème de routage ou de placement de chemins pour connecter les différentes VNFs entre elles [DWARAKI et WOLF, 2016].

[MANN, 2015] propose un survey très complet des différentes approches qui ont été mises en œuvre. Le problème du placement de chaînes de VNFs est aussi désigné par le terme VNF-FG Embedding. Il s'agit en fait de mapper le graphe VNF-FG sur le graphe de l'infrastructure en respectant les objectifs et contraintes.

[LANGE et collab., 2017] proposent *MO-VNFCP* une approche du placement de VNF-FG basé sur une méta-heuristique de simulated-annealing. L'approche délivre un ensemble de solutions pareto-optimales par rapport aux métriques proposées c'est-à-dire un Pareto front.

L'approche proposée permet de placer simultanément dans le réseau un ensemble de

VNF-FGs et aussi de réaliser des ré-optimisations dans la suite de la gestion courante du réseau.

Les contraintes prises en compte, sont des contraintes de latence et de bande passante nécessaires pour le service réseau tandis que les auteurs cherchent à optimiser ces deux métriques et des objectifs de coûts, de ressources consommées dans l'infrastructure par les VNFs, ainsi que le nombre d'instances de chaque type de VNFs placées dans le réseau.

Les deux métriques réseaux considérées sont la latence et la bande passante. Les métriques attachées aux nœuds sont la puissance de calculs en nombre de cœurs, la quantité de mémoire vive et de mémoire de stockage. Les objectifs à minimiser sont le nombre de sauts pour parcourir la VNF-FG, le délai de parcours de la VNF-FG, le nombre de VNFs et la capacité de CPU consommée globale qui dépend du type de VNFs.

Le problème est modélisé comme un ensemble de demandes de trafic de caractéristiques données (source, destination, délai, bande passante) et pour un VNF-FG donné avec des types de VNF données. Un modèle simplifié effectue d'abord un placement approximatif des VNFs sur les nœuds qui respectent les contraintes. Puis la deuxième étape est une recherche de plus courts chemins afin de minimiser la latence pour relier les différentes VNFs et construire le VNF-FG. Enfin un calcul du nombre de VNF à instancier par rapport à la demande est réalisé en utilisant une heuristique approximative de résolution d'un problème de bin packing appliqué sur la bande passante disponible.

Pour obtenir des solutions proches du front de Pareto optimal, les auteurs appliquent une méta-heuristique de simulated annealing paramétrée de manière adéquate. La méta-heuristique explore simultanément plusieurs placements de manière à construire un front de Pareto. Différentes stratégies sont mises en œuvre pour déterminer les solutions initiales:

- purement aléatoire,
- avec une instanciation courte de la méta-heuristique permettant une première optimisation,
- avec une heuristique qui détermine le placement sur le plus court chemin,
- avec une heuristique qui place un nombre minimum de VNF sur des nœuds qui maximisent la betweeness centrality dans le graphe (BRANDES [2001]).

L'évaluation est basée sur trois indicateurs distincts, l'hyper-volume, l'indicateur Epsilon basé sur la notion de dominance qui permet de comparer directement des sous ensembles de solutions [OSMAN et collab., 2006], et enfin un indicateur qui permet d'évaluer la distance au point idéal de l'espace des objectifs.

Les auteurs présentent des résultats pour des problèmes avec des tailles de VNF-FG importantes, jusqu'à 15 VNF et des réseaux de tailles relativement importantes.

L'algorithme est exécuté en parallèle sur une machine multi-cœurs. Les résultats montrent que l'algorithme converge vers un front de Pareto en environ 2 secondes, l'évolution qui suit étant relativement mineure.

La méta-heuristique est comparée à une méta-heuristique de référence et donne des résultats similaires en quelques secondes pour les différents indicateurs choisis.

[ADDIS et collab., 2015] proposent un modèle de programmation linéaire en nombres entiers du problème de placement de VNF-FG.

Le modèle cherche à optimiser deux objectifs différents, éventuellement contradictoires. Il y a un objectif d'ingénierie de trafic qui cherche à minimiser l'occupation du

lien le plus chargé, et il y a un objectif de coût qui vise à minimiser le coût total de la virtualisation en minimisant le nombre de cœurs mobilisés par l'instantiation des VNFs.

Les deux objectifs sont ordonnés pour tester deux scénarios qui priorisent l'un ou l'autre des objectifs.

C'est une stratégie pour traiter des scénarios multi-objectifs avec des techniques de programmation linéaire. On cherche la meilleure valeur pour un objectif donné et celui-ci étant fixé on parcourt l'espace de recherche pour le second en cherchant à le maximiser. Cette stratégie n'est cependant pas nécessairement optimale.

De plus le problème est décomposé en plusieurs étapes qui correspondent à des versions simplifiées (sans latence, sans compression \décompression des flux au niveau des VNFs) jusqu'au modèle complet, chaque étape fournissant des entrées à la suivante.

Dans le modèle, deux régimes de latences sont pris en compte pour les VNFs sous forme de profils, un profil fastpath et un modèle lowpath.

Le profil correspond à un régime de latence en fonction de la charge de trafic qui est déterminé par les moyens logiciels mis en œuvre.

De plus, certaines VNFs peuvent aussi modifier le débit du flux entrant en appliquant des politiques de compression décompression des flux.

L'implémentation a été réalisée sur le solveur CPLEX avec un modèle réseau type qui est structuré en un réseau cœur, un réseau de collecte et un réseau d'accès de caractéristiques différentes. Les capacités de liens introduisent un risque de saturation au niveau du réseau d'agrégation principalement. Ils sont plutôt sur-dimensionnés au niveau du cœur.

Le dimensionnement de la couche NFV-I est réalisé de manière à ce que la moitié des requêtes de VNFs individuelles puissent être satisfaites.

L'analyse des résultats est réalisée sur la base de différentes matrices de demandes produites aléatoirement et couvrant les trois métriques suivantes: le coût de déploiement en terme de cœurs mobilisés, la répartition du trafic, la latence globale moyenne obtenue. Le temps de calcul global pour analyser un scénario est de 15 mn mais la topologie détaillée du réseau testé n'est pas donnée ce qui est dommage, on peut dans ce cas supposer qu'elle est très petite.

Les résultats obtenus et leur analyse qualitative montrent l'intérêt de la modélisation d'un coût de déploiement pour obtenir une bonne répartition des VNF-FG dans le réseau.

[CARPIO et collab., 2017] mettent en œuvre une approche MILP, une approche aléatoire et surtout un algorithme génétique avec plusieurs composantes imbriquées pour traiter le problème du placement VNF-FG et de la réplication des VNFs intermédiaires pour réaliser un bon équilibre du trafic.

L'hypothèse est que l'on peut disposer dans le réseau des nœuds disposant de capacités de stockage qui peuvent héberger des VNFs pour assurer la répartition du trafic dans le réseau. La VNF d'entrée est supposée non réplicable et est disposée dans un datacenter à l'entrée du réseau.

Le modèle de VNF-FG de référence considéré est un réseau mobile virtualisé avec ses différentes entités: eNodeB, S-GW, P-GW, et entitées de traitement des paquets (Load balancer, NAT ...) avant sortie vers l'internet.

Le modèle est composé d'un modèle d'allocation de ressources et d'un modèle d'allocation de trafic. L'objectif est de minimiser un coût d'utilisation des liens qui augmente exponentiellement quand le taux d'occupation se situe entre 60% et 100%.

La méta-heuristique est composée d'un algorithme génétique de calcul de chemins, d'un algorithme génétique d'allocation de ressources, et d'un algorithme génétique pour la réplication des VNFs. Ces différents modules sont composés entre eux et itérés pour

trouver des solutions proches du front de Pareto. L'analyse des résultats montre des résultats presque optimaux pour de petites instances de réseaux et bien meilleurs que l'approche aléatoire introduite pour disposer d'une référence de comparaison.

Une limite de la méta-heuristique proposée est le temps de calcul non négligeable, ce qui peut limiter en pratique le déploiement. Par exemple, le temps de calcul présenté dans [CARPIO et collab., 2017] est compris entre 100 s et 700 s même dans les réseaux de taille modeste où le nombre moyen de nœuds varie de 20 à 100 .

Conclusion sur le placement de chaînes de VNF

Dans cet état de l'art sur le placement de chaînes de VNF, nous avons présenté l'architecture ETSI-NFV qui est l'architecture fonctionnelle de déploiements de VNF-FG.

Puis nous avons analysé quelques exemples de traitement du problème de placement de chaînes de VNF soit en utilisant un modèle MILP soit en utilisant des méta-heuristiques, notamment des AEMO.

2.4 Algorithmes évolutionnaires

Dès l'avènement de l'informatique est venue l'idée de simuler la théorie de l'évolution pour mieux la comprendre. Cette démarche a suscité en parallèle l'émergence d'une catégorie d'algorithmes stochastiques très efficaces pour résoudre des catégories de problèmes variés en s'inspirant du modèle de l'évolution Darwinienne. Le principe de ces approches est calqué sur les principes de bases issus de la théorie de l'évolution qui sont:

- La variation aléatoire des caractéristiques individuelles entre les générations,
- le fait que des caractéristiques peuvent être héritées,
- Un processus de sélection des individus d'une population suivant leurs aptitudes (fitness) en relation avec leur environnement pour survivre et se reproduire.

Après la publication de l'ouvrage de [GOLDBERG, 1989], d'excellentes méta-heuristiques basées sur la structure de ces algorithmes ont été obtenues dans les années 2000 . D'une manière générale ce type d'algorithme s'inscrit dans les approches multiples de résolution de problèmes que l'on appelle bio-mimétiques.

Il s'agit finalement d'imiter certains processus du vivant pour résoudre des problèmes. Les récentes avancées du machine learning basées sur des structures informatiques imitant des réseaux de neurones en sont un exemple incontournable.

Les algorithmes évolutionnaires appartiennent aussi à une classe plus générale d'algorithmes qu'on appelle stochastiques dans le sens où ils utilisent des mécanismes aléatoires dirigés pour explorer l'espace de recherche. On peut aussi citer dans cette catégorie [BLUM et ROLI, 2003]:

- les approches basées sur la méthode de Monte-Carlo,
- les algorithmes de recuit simulés (simulated annealing),
- les algorithmes de tabou search,
- les algorithmes de colonies de fourmis,
- les algorithmes basés sur les essaims particulaires,
- les algorithmes GRASP.

Le champ d'applications de ce type de techniques est devenu très varié, de l'optimisation, en passant par l'apprentissage avec la recherche de solutions originales à des problèmes variés d'ingénierie.

L'apogée des travaux se situe dans les années 2000-2010 avec la proposition de métaheuristiques performantes. On peut faire une distinction entre trois catégories:

- les algorithmes génétiques (AG): qui désigne des algorithmes destinés à l'optimisation et manipulant des structures de données qui sont sous forme de mots binaires qui correspondent un peu aux génomes des individus dans le processus du vivant. Dans les AGs les opérateurs de croisement et de mutations sont standardisés.
- Les algorithmes évolutionnaires (AE) dont la structure est beaucoup plus large, flexible et adaptable, y compris pour ce qui concerne la structure des données manipulées, les opérateurs mis en œuvre et le champ d'application (apprentissage supervisé par exemple).

- La programmation évolutionnaire (PE) initiée par les travaux de [KOZA, 1992] qui vise à faire évoluer selon les mêmes principes des structures de données représentant des programmes modélisés par exemple sous forme d'arbres. C'est à l'exécution du programme que dans ce cas le résultat est évalué pour introduire ensuite des modifications et améliorer la structure qui est l'objet du problème. On range dans cette catégorie des techniques visant à faire de l'apprentissage non supervisé et des approches visant à optimiser des structures complexes comme des réseaux de neurones.

Pour éviter trop de confusion, nous utilisons ici le terme algorithme évolutionnaire qui a l'avantage d'être assez général et qui reflète la flexibilité qu'offre ce genre d'algorithmes.

2.4.1 Vocabulaire

Il ne s'agit pas d'un verbiage [6] mais d'un vocabulaire issu strictement de la théorie de l'évolution qui permet de bien résumer les concepts mis en œuvre dans ce genre d'algorithmes. On désigne par:

- Individu: une structure de données qui représente une solution de problème bonne ou mauvaise, valide ou non valide du problème posé.
- Parents: Les individus peuvent être des parents auxquels vont être appliqués des opérateurs de re-combinaisons et de mutations pour produire des enfants.
- Génome: structure de données représentant un individu et adaptée à la résolution du problème, dans le cas des algorithmes génétiques, il s'agit d'un mot binaire. Mais toutes sortes de structures sont possibles.
- Phénotype: correspond à des caractéristiques observables du génome qui peuvent par exemple être reliées à la fitness (une sous séquence de bits aux propriétés particulière par exemple).
- Population: un ensemble d'individus, la taille de la population est un des paramètres important d'un AE pour assurer un échantillonnage satisfaisant de l'espace des solutions
- Fitness: aptitudes attachées aux individus, c'est la valeur de la ou des fonctions objectives pour lesquelles on cherche des solutions les meilleures possibles.
- Pression de sélection: l'intensité avec laquelle on sélectionne les individus au fil des itérations. Elle est déterminante pour une bonne convergence de l'algorithme. La pression de sélection doit permettre la convergence de l'algorithme en un temps acceptable en évitant les minimums locaux. Si elle est trop forte, l'algorithme peut converger trop vite vers un minimum local. Si elle est trop faible l'algorithme peut mettre trop de temps pour converger.
- opérateurs de variations (mutation et croisement): ce sont les opérateurs qui imitent les mécanismes de l'évolution darwinienne en introduisant de la variation (mutations) dans les individus ou en définissant un mécanisme d'héritage (croisement) avec une part d'aléatoire, des bonnes caractéristiques de parents pour produire des enfants.

[6] en référence à la remarque désagréable d'un reviewer

- Opérateur de sélection: le mécanisme par lequel on va privilégier les meilleurs individus dans la population au fil des itérations de l'algorithme.

- Evaluation: le mécanisme d'évaluation de la fitness des individus de la population permettant d'appliquer l'opérateur de sélection. Il correspond à l'évaluation de la ou des fonctions objectives du problème.

2.4.2 Structure générale des algorithmes évolutionnaires

La structure est illustrée dans le diagramme 2.9.

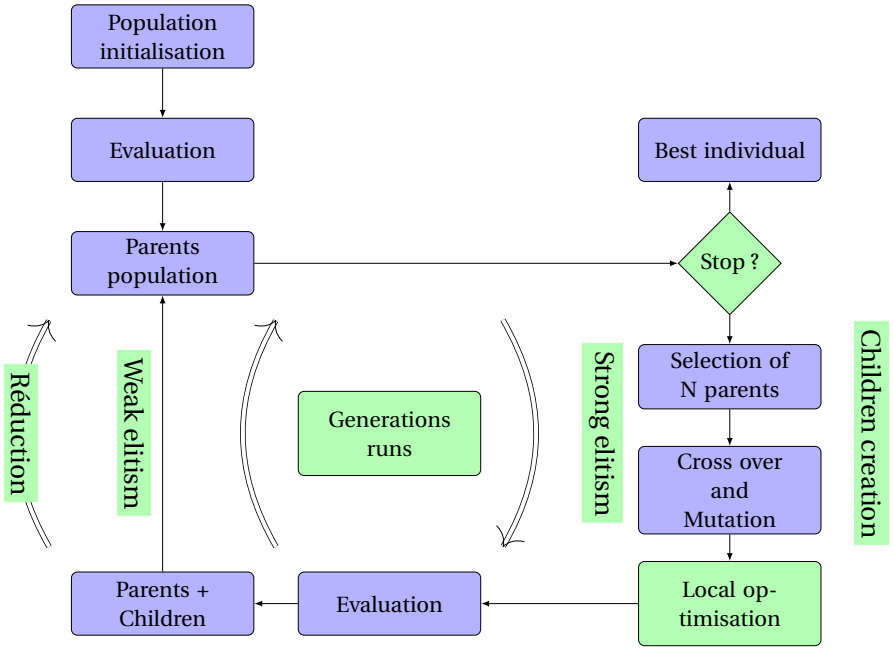


Figure 2.9 – Structure générale d'un algorithme évolutionnaire

Initialisation

A l'étape d'initialisation, une population d'individus représentant des solutions possibles du problème est créée à partir d'un mécanisme de tirage aléatoire. L'objectif est de bien balayer l'espace de recherche.

La taille de la population est un paramètre important qui peut varier de manière adaptative mais qui dans les versions classiques est maintenue constante au fil des itérations de l'algorithme.

L'idée est de disposer au départ d'un éventail de solutions suffisamment diverses pour ne pas converger vers un minimum local. La taille de la population dépend beaucoup des caractéristiques du problème, des caractéristiques des différents opérateurs de l'algorithme et est généralement définie de manière empirique. Elle dépend aussi de la puissance de calcul disponible.

Il existe cependant des publications comme celle de [A. DIAZ-GOMEZ et HOUGEN, 2007] qui s'inspirent de l'idée de disposer d'une population suffisamment diverse pour estimer la taille de la population.

Globalement comme dans d'autres étapes de l'algorithme, l'enjeu est de disposer d'une diversité suffisamment grande de solutions possibles dans l'espace des états du

problème, de manière à éviter de converger vers un minimum local. Il existe aussi la notion de taille efficace d'une population utilisée par les biologistes [7].

Evaluation

Dans l'étape d'évaluation, les individus sont ensuite évalués et leur fitness (adaptation) c'est à dire la valeur des fonctions objectives qui leur correspondent sont calculées et affectées aux individus. La nouvelle population évaluée correspond alors à une population de parents. Parmi ces parents, un des individus peut déjà correspondre à une solution acceptable du problème et dans ce cas il est possible de stopper l'algorithme et de retenir cette solution.

Sélection

S'il n'y pas de solutions convenables, on passe à l'étape de sélection. La sélection simule la pression de l'environnement sur les individus pour tendre à retenir les mieux adaptés. Elle permet de favoriser certains individus qui vont être retenus dans l'étape de recombinaison (cross-over ou mating) qui suit pour produire une progéniture. Elle intervient aussi dans le renouvellement de la population.

Différents opérateurs de sélection existent et ont fait l'objet de comparaisons comme dans les travaux de [BLICKLE et THIELE, 1995]. L'opérateur basique inspiré directement de la théorie de l'évolution darwinienne, consiste à associer aux individus une probabilité d'être retenus proportionnelle à leur fitness (roulette wheel selection). La probabilité p_i d'un individu i d'être sélectionnée est donc proportionnelle à sa fitness f_i, comme dans l'équation 2.2.

$$p_i = \frac{f_i}{\sum_{j=1}^{\mu} f_j} \tag{2.2}$$

Soit μ la taille de la population, k le nombre de fois que l'on applique l'opérateur de sélection pour sélectionner k individus, on a une distribution binomiale $B(k, p_i)$ qui décrit la probabilité de tirer un certain nombre de fois l'individu i sur les k tirages, dont l'espérance vaut $k \cdot p_i$ et la variance $k \cdot p_i \cdot (1 - p_i)$.

Si la taille de la population (μ) est suffisamment grande, la variance est de l'ordre de $k \cdot p_i$. Celle ci peut être élevée si la fitness de l'individu est grande. De ce fait, l'opérateur de sélection peut récupérer des individus avec une fitness peu élevée en manquant des individus avec une bonne fitness.

Plus généralement, le problème de cette méthode réside dans le fait que la pression de sélection est fonction du paysage de fitness, et diffère suivant le problème. Elle peut aussi induire la convergence vers des minimums locaux.

Pression de sélection Pour définir une métrique de pression de sélection notée p_s, on part de l'espérance du "meilleur individu" de fitness $\widehat{f}$ et on divise par la fitness moyenne $\overline{f}$. [PETROWSKI et BEN HAMIDA, 2017]. L'idée est de réaliser μ opérations de sélections parmi une population de μ individus et de mesurer le nombre d'individus avec la meilleure fitness susceptible d'être obtenue (ou l'espérance de cette variable aléatoire). . Si $p_s = 1$, tous les individus ont une chance égale d'être sélectionnés, et il n'y a pas de

[7] https://www6.inra.fr/angr/content/download/3191/32485/version/1/file/R2GA_2016_FHospital_v3.pdf

pression de sélection ce qui est intuitivement cohérent.

$$p_s = \frac{\widehat{f}}{\overline{f}} \tag{2.3}$$

Une des manières de résoudre le problème de la variance a été d'introduire une sélection basée sur l'ordonnancement des individus selon leur fitness plutôt qu'une sélection proportionnelle à la valeur de la fitness. L'efficacité de l'opérateur de sélection ne dépend alors plus du paysage de la fonction objectif mais simplement d'un classement et on obtient alors une fitness qui est régularisée et beaucoup plus facile à manipuler. Elle est cependant coûteuse en calcul car il faut ordonner les individus entre eux.

Finalement, la solution la plus couramment utilisée est un mécanisme simple de tournois (selection tournament) [L. MILLER et E. GOLDBERG, 1995]. Il consiste à tirer au hasard (avec remise dans la population de parents) p individus et de retenir le meilleur au sens de leur fitness. On réalise autant de tournois que l'on veut sélectionner d'individus. Plus p est élevé plus la pression de sélection est importante, puisque la probabilité d'obtenir un individu avec une fitness élevée augmente.

En pratique des valeurs au delà de $k = 7$ [XIE et ZHANG, 2013] n'ont pas d'intérêt car la pression de sélection devient alors trop forte. Avec ce mécanisme, la probabilité de tirer le meilleur individu à chaque essai est simplement $\frac{p}{\mathrm{N}}$, où N est la taille de la population.

Si on réalise $k = \mathrm{N}$ tournois pour tirer N individus, puisqu'on effectue un tirage avec remise, $\frac{p}{\mathrm{N}} \cdot k$ vaut p et on peut espérer trouver p fois le meilleur individu dans les individus sélectionnés. La valeur de p correspond bien donc à une définition de la pression de sélection.

Il existe des variantes qui permettent d'affiner la pression de sélection, par exemple une méthode simple consiste à ne tirer le meilleur individu parmi les p qu'avec une probabilité p_e, ce qui permet d'affiner le curseur de la pression de sélection. Dans ce cas la pression de sélection vaut $p_s = p \cdot p_e$ et peut-être mieux ajustée. D'autres définitions de la pression de sélection existent.

Opérateurs de variation

Le but des opérateurs de variation est de générer une "progéniture" avec autant que possible une meilleure fitness pour les individus de la progéniture en explorant au mieux l'espace de recherche [SIEW MOOI et collab., 2017]. Il y a deux opérateurs calqués sur les mécanismes du vivant.

L'opérateur de mutation permet de modifier de manière stochastique les individus et d'introduire ainsi de la variabilité dans la progéniture.

L'opérateur de recombinaison (cross-over) combine plusieurs parents pour créer de nouveaux individus. La recombinaison peut s'appliquer à deux parents comme dans le vivant mais on peut imaginer des opérateurs de recombinaison qui agissent sur $k \geq 2$ parents voire s'appliquent sur l'ensemble de la population.

La manière dont ces opérateurs sont définis et agissent dépend fortement de la structure de données qui représente les individus et aussi des caractéristiques du problème. Ils ne peuvent pas être complètement généralisés même s'il existe certains opérateurs standards.

Ces opérateurs contribuent à réaliser le compromis "exploration", c'est-à-dire la capacité a bien balayer l'espace de recherche, contre "exploitation", c'est-à-dire la capacité à identifier les meilleures solutions dans une zone donnée de l'espace de recherche qui permet de garantir que l'algorithme va identifier de bonnes solutions.

Le tuning des opérateurs de variation (taux de mutation et de recombinaison par exemple) est essentiel pour les performances de l'algorithme.

Opérateur de recombinaison ou de croisement Bien qu'il existe quelques opérateurs de croisement standards pour des problèmes continus ou discrets codés sous forme de mots binaires, la conception de ces opérateurs n'est pas soumise à des règles précises.

L'idée générale est que la progéniture des k parents doit statistiquement avoir une meilleure fitness que celle des parents et doit donc d'une manière ou d'une autre récupérer des propriétés favorables dans le génome des parents.

Il y a quelques principes à suivre cependant mais qui ne sont absolument pas intangibles:

- Le croisement d'individus identiques doit donner le même individu,
- Le mécanisme est stochastique, c'est-à-dire que répéter sur les même parents peut donner des résultats différents,
- Des parents "proches" (au sens d'une distance à définir), produisent plutôt des individus qui leur sont proches,
- Un taux de recombinaison détermine une proportion d'individus croisés dans la population.

Dans le cas où la structure d'un individu est représentée sous forme d'un mot binaire, il existe des opérateurs de cross-overs simples qui peuvent être utilisés entre deux parents mais qui ne sont pas nécessairement adaptés aux caractéristiques du problème. Ils peuvent même être totalement inefficaces.

Nous en décrivons trois ci-dessous à titre d'exemple.

- Le single point cross-over où l'on définit un point de coupure sur les mots binaires des deux individus et ou l'on échange un des segments.
- Le two "point" cross-over, où l'on définit deux points de coupures et où l'on échange le segment du milieu.
- le uniform cross-over, où des segments multiples des deux mots binaires représentant chaque parent peuvent être échangés avec une probabilité donnée. Les mots ou bits échangés sont définis par un template.

Opérateur de mutation En principe, le but de l'opérateur de mutation est de maintenir de la diversité dans les solutions explorées par l'algorithme au fil des itérations, il contribue donc à la fonction d'exploration de l'espace des solutions.

Pour cela il introduit de manière aléatoire de la variation dans les individus enfants généralement mesuré par un taux de mutation.

Si la structure de données est un mot binaire, une manière de procéder et de tirer au hasard k bits qui vont être inversés au hasard. Une autre manière de procéder est de faire en sorte que chaque bit du mot puisse être inversé avec une certaine probabilité, plutôt de faible valeur pour ne pas trop dégrader les solutions trouvées.

D'autres stratégies ont été imaginées en fonction de la structure de données qui représente les individus (exemple d'un vecteur de réels) et dans ce cas aussi il est possible d'innover en fonction de la structure des individus et des particularités du problème.

Taux de mutation Dans un algorithme évolutionnaire, le paramétrage du taux de mutation est un point délicat [BÄCK, 1993].

On peut cependant noter un résultat des biologistes dans leur travaux de modélisation de l'évolution. Pour qu'une population composée de génomes (appelée ici quasi-specie) puisse converger vers des individus avec la fitness optimale il faut que le seuil de mutation soit plus petit que l'inverse de la longueur du génome [NOWAK, 2007].

En résumé, ce qui compte c'est que le taux de mutation ne soit pas trop élevé pour permettre de fixer des individus avec une meilleure fitness grâce à la sélection.

Cette règle est valable en général sauf pour certains "paysages" de fitness pathologiques.

Ainsi pour un génome de taille 100 le taux de mutation ne devra pas dépasser 1%.

De manière assez intuitive si le taux de mutation est trop élevé l'algorithme ne pourra pas trouver la solution optimale, car il explorera de manière trop intense l'espace de recherche.

Cette règle tirée des simulations des biologistes correspond de manière surprenante à la règle empirique de mise en œuvre dans la conception des algorithmes génétiques.

Discussion sur les AE à mutation seule et sur l'utilité de l'opérateur de recombinaison

Il existe peu de débats dans la communauté des informaticiens sur la pertinence des opérateurs de mutation et de recombinaison dans les algorithmes évolutionnaires. Pourtant, cette problématique est largement étudiée dans le contexte du vivant par les biologistes du fait des différentes stratégies adoptées par le vivant.

Pour les biologistes, la recombinaison des individus via la reproduction permet de favoriser la co-adaption en accélérant le cumul de différents avantages évolutifs dans la progéniture. D'un autre côté elle ralentit le renouvellement des individus dans la population puisqu'il faut appairer des individus pour produire une descendance !

Généralement dans la description des AE, on présente l'opérateur de mutation comme en charge de la partie exploration (de l'espace de recherche) par opposition à la partie exploitation portée par l'opérateur de re-combinaison. Si l'on creuse un peu, cette vision semble cependant un peu insuffisante.

Pour que l'opérateur de recombinaison permette une bonne exploitation, il faudrait qu'on ait la garantie qu'il permette de produire une meilleure progéniture avec une probabilité suffisante. Cela dépend bien sûr de sa conception mais n'a rien d'évident et dépend fortement du problème.

De plus on pourrait aussi placer la partie exploitation de l'algorithme dans le processus de sélection qui est chargé de retenir les meilleurs individus.

Enfin on pourrait aussi placer l'exploration en partie dans l'opérateur de recombinaison dans la mesure où lui même fait appel à des mécanismes aléatoires.

La mise en œuvre d'une étape de recombinaison dans un AE s'inspire du modèle du vivant. On peut se poser cependant la question de l'intérêt ou de l'utilité de cette étape dans les algorithmes évolutionnaires.

Pourquoi ne pas utiliser des algorithmes où le seul opérateur de variation est l'opérateur de mutation comme d'ailleurs cela existe dans le vivant au niveau cellulaire par exemple ?

Dans le cas du vivant, [FISHER et BENNETT, 1999] montrent que la sexualité semble constituer un avantage évolutif important, en facilitant la transmission de caractères évolutifs avantageux, dans les conditions où les populations sont très homogènes génétiquement tout en facilitant l'évacuation de caractères désavantageux si au contraire il y a une

forte diversité génétique.

Dans le cas de la mise en œuvre d'algorithmes évolutionnaires, on a la latitude de mettre en œuvre ou pas un opérateur de recombinaison. Pour que l'opérateur de recombinaison ait un intérêt, il faut qu'il soit efficace c'est-à-dire que le mécanisme permette de transmettre avec une certaine probabilité une partie du génome qui contribue à améliorer la fitness d'un individu à la progéniture.

La définition de l'opérateur de recombinaison risque d'être problème dépendant sinon il sera peu efficace voire inefficace.

Cependant, si l'opérateur de recombinaison est efficace il permet d'accélérer la convergence de l'algorithme en réduisant le nombre de générations. Il faut aussi qu'il soit peu coûteux en calcul car sinon l'avantage de réduire le nombre de générations serait annulé.

Dans le contexte du "computationnal learning" et de la théorie Probability Approximately Correct (PAC) de l'apprentissage, [VALIANT, 1984] propose un modèle de l'approche algorithmique évolutive, "the evolvable" comme une forme réduite d'apprentissage. Ce modèle est équivalent au modèle CSQ ("Correlational Statistical Query") en apprentissage [FELDMAN, 2008].

Ce modèle n'inclut qu'un opérateur de mutation sans recombinaison. Il est ensuite étendu par [KANADE, 2011] avec la recombinaison. Les auteurs montrent que sous certaines conditions le nombre de générations peut être drastiquement réduit en passant de n générations à un facteur proportionnel à $\log(n)^2$. Ce résultat même si il n'est peut être pas généralisable suggère qu'on peut considérablement augmenter les performances d'un algorithme évolutionnaire si on est capable d'introduire le bon opérateur de recombinaison.

D'un autre côté, un algorithme avec un simple opérateur de mutation peut aussi être suffisant si le problème s'y prête.

Enfin, dans le vivant, l'évolution non sexuée existe très significativement par exemple parmi les organismes les plus simples comme les bactéries ou les plantes.

Cela confirme qu'on peut très bien concevoir et modéliser des processus évolutifs en s'inspirant d'un processus de divisions cellulaires et de mutations sans opérateur de recombinaison.

Conclusion sur les opérateurs de variation Dans les AE, l'opérateur de recombinaison a un intérêt si on est capable de le concevoir avec de bonnes propriétés, ce qui est "problem dependent" et difficilement généralisable.

D'autre part, un AE à mutation seule est parfaitement viable à l'image de processus évolutifs non sexués qui se produisent dans la nature.

Dans l'analyse du rôle des opérateurs de variation, on peut aussi remarquer que l'introduction d'un opérateur de recombinaison peut induire un coût en temps de calcul et de nouveaux paramétrages complexes (taille de la population par exemple).

D'après [VALIANT, 2013], la modélisation du vivant comme un processus d'apprentissage suggère que la reproduction sexuée semble permettre de réduire le nombre de générations au prix d'une plus grande taille de la population. C'est un aspect à prendre en compte dans l'introduction d'un opérateur de recombinaison dans un AE.

Elitisme fort et faible

Dans le diagramme 2.9 le mot "élitisme" apparaît à deux étapes. En premier lieu au moment du renouvellement de la population initiale, et aussi au moment de l'étape de variation mutation + croisement).

Dans le premier cas, on parle d'élitisme faible dans la mesure où le scénario de base consiste à remplacer tout ou partie de la population des parents par les enfants, dans le deuxième cas on parle d'élitisme fort.

Dans le premier cas, les modalités de renouvellement peuvent varier et différentes stratégies peuvent être mises en œuvre susceptibles de renforcer ou de réduire l'élitisme.

- Remplacement générationnel: on remplace l'ensemble de la population de parents par la population d'enfants, sans garanties que la population sera effectivement meilleure,

- Elitisme fort $\mu + \lambda$: [COELLO et collab., 2006] On conserve un ou plusieurs (μ) individus de la population courante, les meilleurs et on remplace le reste par λ enfants, avec un risque de convergence prématurée ,

- Steady state: [DAHAL et MCDONALD, 1997] On produit un enfant à chaque itération $\mu = 1$ qui remplace le plus mauvais des parents ou le plus mauvais de la population, ou un individu de la population sélectionné suivant un certain critère (anti sélection par tournoi par exemple).

Mais dans certaines conditions par exemple pour des algorithmes que l'on cherche à paralléliser, il peut être handicapant de mettre en place de l'élitisme au niveau du renouvellement de la population car cela suppose d'effectuer des comparaisons entre individus ce qui empêche le parallélisme.

Une manière de résoudre ce problème consiste dans ce cas à faire de l'élitisme au niveau de la génération des enfants.

On peut réaliser une sélection après la génération des enfants (mutation et recombinaison) en choisissant de retenir le meilleur des individus parmi le groupe constitué des parents et des enfants plutôt que de retenir nécessairement les enfants. Ensuite on peut faire un un simple remplacement générationnel pour le renouvellement de la population.

On aura ainsi déplacé l'étape de sélection au niveau de la création des enfants, ce qui ne gênera pas le parallélisme. On aura par contre ici un élitisme fort car cette méthode peut être très sélective.

Optimisation locale ou hybridation

Particulièrement dans les problèmes d'optimisation combinatoire, il peut être intéressant d'intégrer une fonction d'optimisation locale qu'on appelle aussi hybridation. Il y a deux approches possibles. elle apparaissent dans le schéma 2.9 après l'application des opérateurs de variations.

- Améliorer directement la population d'enfants en introduisant dans chaque génération un opérateur qui va appliquer un algorithme d'optimisation "local" sur chaque enfant. On appelle cela le lamarckisme ou encore l' approche épigénétique.

  Cela correspondrait dans le processus du vivant à une amélioration des individus au cours de l'existence et à une transmission des caractères acquis. La théorie de Lamarck antérieure au darwinisme est exploitée dans le papier de [HOLZINGER et collab., 2014] où les AE sont utilisés pour du machine learning. Le risque de cette approche est cependant de trop accélérer la convergence vers un minimum local.

- Evaluer le potentiel de la population d'enfants (baldwinisme) en calculant une optimisation locale mais sans modifier l'individu. Baldwin avait entre autre remarqué qu'un avantage évolutif (par exemple un gros cerveau) pouvait être handicapant au départ mais constituerait un avantage considérable sur le long terme parce qu'offrant un fort potentiel (apprentissage) [SIMPSON, 1953].

  Ici l'idée est de favoriser la sélection des individus à fort potentiel dans l'étape de remplacement de la population de parents par des enfants mais sans modifier leur génome. Le potentiel est évalué avec une optimisation locale comme dans le lamarckisme. L'avantage de cette approche est qu'elle peut permettre de limiter le risque de convergence prématurée.

Une idée qui vient naturellement à l'esprit est d'intégrer dans l'opérateur de recombinaison, une fonction d'optimisation qui peut être vu comme un mécanisme d'hybridation local proche du Lamarckisme. Dans la mesure où l'on attend de l'opérateur de recombinaison qu'il améliore statistiquement les enfants par rapport aux parents.

Une manière de faire consiste donc à chercher à optimiser le résultat à l'intérieur de cet opérateur.

2.4.3 Importance de la structure de données dans les AE

L'implémentation d'un AE dépend beaucoup de la structure de données qui représente un individu c'est-à-dire de la manière de modéliser une solution du problème.

Il y a de multiples manière de modéliser. On peut par exemple se contraindre à utiliser un mot binaire comme dans certains modèles des biologistes [NOWAK, 2006] mais des opérateurs standards de variation ne seront pas nécessairement efficaces dans ce cas.

En fonction de la typologie du problème par exemple un problème de routage ou un problème de placement de contrôleurs, le choix d'une structure de données peut faciliter ou complexifier la définition des opérateurs de variations.

Pour une structure de données choisie, un opérateur de mutation mal adapté peut produire des solutions non viables. Au contraire une structure de données bien choisies peut permettre de définir facilement les opérateurs de variations adéquats et de les rendre immédiatement efficaces.

Exemple du regroupement (clustering)

L'utilisation des algorithmes génétiques standards n'est pas particulièrement efficace pour faire du regroupement de données principalement du fait de la structure de données binaire adoptées pour représenter des individus, d'autres modèles ont cependant donné de bons résultats.

Le terme GGA (Grouping Genetic Algorithm) a été imaginé par FALKENAUER [1993]. L'idée est de définir des structures de données représentant les individus plus adaptées que de simples listes de nombres entiers où des mots binaires représentant l'affectation des données aux clusters comme en Figure 4.24.

Ces structures de données doivent permettre de définir des opérateurs de variation plus efficace pour obtenir des AE efficaces pour des tâches de regroupements.

L'auteur montre en particulier que la structure de données classique à savoir une liste d'entiers de longueur fixe n'est pas adaptée pour concevoir un algorithme génétique efficace. En effet sur une telle structure un opérateur de mutation standard qui opère par exemple en remplaçant l'attachement d'un nœud à un groupe par un autre groupe peut

dégrader très fortement la fitness d'un individu. Il suffit par exemple d'imaginer un nœud qui constituerait à lui tout seul un groupe à la suite d'une mutation.

De même, une telle structure ne permet pas de conserver des individus viables si on applique un opérateur de recombinaison qui coupe des segments du génome des parents et les réassemble. Dans ce cas on obtient généralement des individus non viables avec des nœuds répétés dans plusieurs groupes ou des nœuds manquants.

Exemple du routage

Dans le cas du routage la structure de données doit décrire un chemin simple dans le graphe. Un chemin simple est en fait un sous ensemble ordonné de nœuds ou d'arcs contiguës du graphe sans cycles.

Dans ce cas, on peut voir apparaître, du fait des opérateurs de variations, des individus non viables, parce que les nœuds ou les arcs sont non contiguës ou encore parce que les individus comportent des cycles. La question de déterminer l'ordre de parcours doit être inclue dans la structure de données à moins qu'elle ne soit retrouvée à partir de la donnée du graphe lui même.

Si l'on choisit de travailler avec un mot binaire une manière de procéder est de choisir une structure de données ordonnées qui décrit la liste de tous les nœuds ou de tous les arcs du graphe.

Un chemin est alors décrit par l'ensemble des bits de valeurs 1, qui correspondent à des nœuds ou des arcs contiguës dans le graphe.

Un chemin peut être aussi décrit par une liste d'arcs de valeur 1, qui correspondent à des arcs contiguës du chemin.

L'inconvénient de cette structure est qu'elle ne reflète pas l'ordre de parcours des nœuds ou des arcs.

Une telle structure est adaptée cependant à la mise en œuvre d'opérateurs de mutations où de croisement standard. Mais elle implique aussi de manipuler une forte proportion d'individus non viables. Un avantage de cette structure est qu'elle prévient les cycles dans la mesure ou un nœud où un arc du graphe n'apparaît qu'une fois.

Pour ne manipuler que des solutions viables au cours de l'algorithme, une solution est d'introduire une fonction de réparation "reparateur" [MITCHELL et collab.] qui permettra de passer à chaque itération d'individus non viables à des individus viables.

Un autre modèle possible non binaire est une simple description du chemin par la liste des nœuds de transit sous forme d'entiers.

Le problème du routage présente des similitudes avec le problème du voyageur de commerce. En effet pour ce problème on doit aussi manipuler une liste ordonnée de nœuds ou d'arcs. Il nécessite aussi du fait de sa spécificité des opérateurs de croisements spécialement conçus par rapport aux particularités du problème.

Différentes approches sont décrites par [POTVIN, 1996] pour le problème du voyageur de commerce.

Il y a cependant quelques différences importantes par rapport au problème de routage.

Dans le problème Traveller Salesman Problem (TSP), toutes les villes sont représentées dans le génome et une solution du problème est une simple permutation de celles-ci. Dans le problème du routage seul un sous ensemble de nœuds du graphe est solution du problème.

De plus, dans le problème du voyageur de commerce n'importe quelle liste de nœuds est viable puisque le graphe est complet (il existe toujours un chemin entre deux nœuds).

Ce n'est pas le cas pour le routage. Les performances des AE ont beaucoup augmenté sur le problème du voyageur de commerce quand les chercheurs ont imaginé des stratégies efficaces de croisement pour ce problème spécifique.

Ces stratégies efficaces sont généralement inspirées d'un ERO (Edge Recombination Operator) d'après [NAGATA et KOBAYASHI, 1999]. L'idée est de s'intéresser aux arcs des chemins des parents plutôt qu'aux nœuds le long des chemins des parents et de construire une liste de tous les arcs contiguës (ou adjacents) aux chemins décrits par les deux parents à chaque nœud successif d'un des parents (le plus court).

Enfin on construit un nouveau chemin à partir de cette liste en utilisant différentes stratégies. L'idée est globalement de conserver des arcs appartenant aux chemins des deux parents. Ce genre d'approche a permis d'obtenir de bonnes performances pour le TSP avec des AE.

Dans ces approches, l'algorithme est construit autour de mécanismes conçus pour ne manipuler que des solutions viables, ce que la structure de données ne permet généralement pas de réaliser de manière simple.

C'est une approche possible pour le problème du routage plutôt que d'introduire un opérateur de réparation à chaque étape.

Il faut dans ce cas garantir qu'à chaque étape de l'algorithme AE, les procédures de tirage, mutation, sélection sont conçues pour produire des solutions viables. Mais si l'on veut réaliser un solveur un peu générique, il est préférable dans ce cas d'utiliser un "repairer" dont la constitution pourra varier en fonction du problème à traiter.

2.4.4 Algorithmes évolutionnaires multi-objectifs

Nous présentons dans cette partie les principes et les idées mises en œuvre dans les algorithmes évolutionnaires multi-objectifs.

Des survey variés traitent aussi le sujet comme celui de [ZHOU et collab., 2011].

Les approches initiales adoptées pour traiter ce type de problèmes avec des AEs consistaient à traiter chaque objectif séparément et à mélanger les solutions à chaque générations de l'algorithme. La proposition de AEMO de [SCHAFFER, 1985], intitulée VEGA pour Vector Evaluated Genetic Algorithm est la plus connue.

Cet algorithme ne connaît pas la notion de front de Pareto et a l'inconvénient principal de tendre à spécialiser des sous populations par objectifs sans trouver des individus combinant des fitness élevées pour plusieurs objectifs. l'algorithme VEGA est souvent utilisé pour établir des comparaisons avec des algorithmes plus performants.

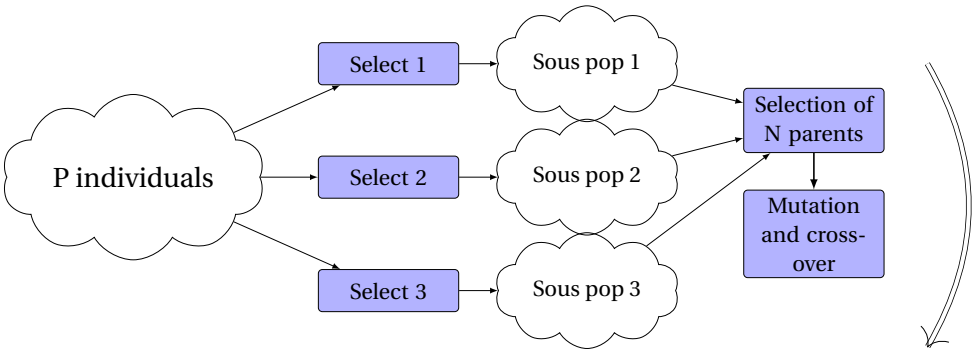


Figure 2.10 – Algorithme VEGA

NSGA II

NSGA II pour "Non dominated Sorting Genetic Algorithm" fait partie des algorithmes évolutionnaires multi-objectifs les plus performants et est décrit de manière très complète par [DEB et collab., 2002b]. C'est un algorithme multi-objectif qui s'appuie sur un élitisme fort c'est-à-dire un remplacement de la population courante par les meilleurs individus choisis parmi la population courante et les enfants.

Il utilise surtout aussi un mécanisme de classification des individus suivant un critère de dominance et suivant un critère de densité (crowding distance) pour assurer une bonne diversité des solutions le long du front de Pareto.

L'approche naïve de classification par Front de Pareto se réalise de la manière suivante. On recherche le front de Pareto de la population, c'est-à-dire les individus qui ne sont pas dominés. Ces individus sont indexés avec l'indice le plus élevé et un front de Pareto est recherché parmi les individus restants pour leur attribuer un indice moins élevé. L'opération est répétée jusqu'à ce qu'il ne reste plus aucun individu non classifiés dans la population. Dans le pire des cas, ce calcul a une complexité en $O(M \cdot N^3)$ ou N est la taille de la population et M le nombre d'objectifs, et en supposant que N fronts successifs soient trouvés.

Cet algorithme a été amélioré dans NSGA II pour réaliser une procédure de classification de complexité en $O(M \cdot N^2)$. Le schéma 2.11 illustre cette notion de classification par front de Pareto sur l'espace des états pour un problème bi-objectifs. Les solutions sont classifiées par des indices un peu comme des pelures d'oignons. L'indice 1 correspond au premier front de Pareto trouvé et les indices décroissent d'une unité à chaque nouveau front.

Un second indice est évalué pour chaque individu, il s'agit d'une "crowding distance" c'est à dire d'une densité normalisée qui correspond à l'hyper-surface délimitée par les deux points les plus proches de la solution selon les différents objectifs. Cette distance permet de distinguer entre des individus qui ont le même indice de dominance. Le schéma 2.12 illustre ce mécanisme.

L'algorithme NSGA II utilise une procédure d'élitisme fort pour renouveler la population. A chaque itération, une fois produite une population d'enfants, la population initiale et la population d'enfants sont fusionnés pour former une population de taille $2 \times N$. La réduction à N pour garder constante la taille de la population se fait en classant les individus par indice de dominance.

Comme les individus d'un même front ne peuvent pas être comparés, ils sont aussi comparés grâce à la crowding distance. Celle-ci permet de manière efficace de bien répartir les solutions trouvées le long du front de Pareto.

Le mécanisme pour choisir les parents est une sélection par tournoi binaire. Mais pour distinguer entre deux parents tirés au hasard du même front de Pareto, la crowding distance est aussi utilisée. La crowding distance a aussi l'avantage de ne pas nécessiter de paramétrage.

Une limite de l'algorithme NSGA II est que la partie évaluation selon la dominance ne peut pas être parallélisée car il faut réaliser une comparaison des individus de la population entre eux.

D'autres algorithmes comme SPEA II de [ZITZLER et collab., 2001] concurrent de NSGA proposent des stratégies pour résoudre cette difficulté en s'appuyant sur la notion d'archives et en ne calculant pas de manière exacte le rang des individus, et en le comparant simplement au contenu de l'archive.

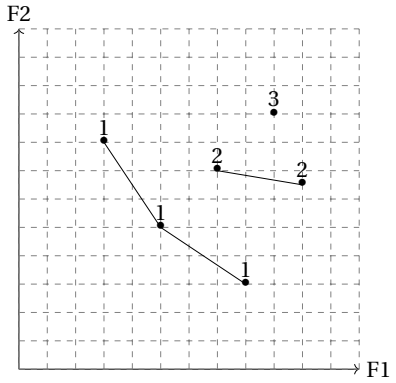


Figure 2.11 – Classification par dominance

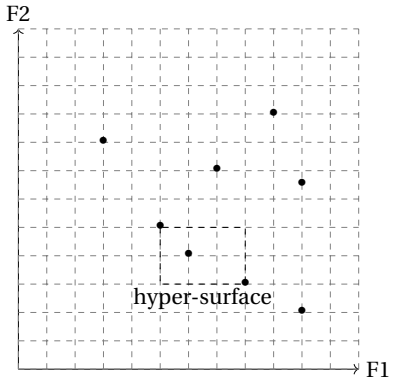


Figure 2.12 – Classification par densité

2.4.5 Parallélisme

Différentes stratégies possibles

Les algorithmes évolutionnaires se prêtent bien au parallélisme. En effet la plupart des étapes peuvent être traitées en parallèle. C'est donc une option qu'il faudrait employer au maximum bien que cela ne soit pas encore généralement le cas aujourd'hui.

L'étape d'initialisation est facilement parallélisable, chaque processeur exécutant ses instructions pouvant créer un individu au hasard.

L'étape d'évaluation est parallélisable, en effet elle s'effectue par individu, elle peut donc être réalisée séparément par différent processeurs. Le critère d'arrêt est parallélisable.

L'opérateur de sélection des parents est parallélisable puisqu'il procède généralement par tirage aléatoire, et que c'est un tirage aléatoire avec remise dans la population.

Les opérateurs de mutation et de croisement peuvent être parallélisés sur les différentes instances d'individus ou de parents. L'évaluation qui suit la création des enfants est parallélisable.

La principale difficulté se situe au niveau du renouvellement de la population, à partir de la population de parents et d'enfants, qui va être réduite par sélection (étape de réduction). En fonction des stratégies d'élitisme adoptées, cette étape peut nécessiter de comparer les individus, sinon le fait de sélectionner en parallèle va tendre à sélectionner les même individus. Elle n'est donc pas dans ce cas parallélisable.

Bien sûr cette question ne se pose pas si l'on effectue un renouvellement complet de la population, sans élitisme au niveau du renouvellement. Dans ce cas il n'y a pas de difficulté.

Une autre manière de faire consiste à introduire l'élitisme au niveau du choix entre N parents et enfants plutôt qu'au niveau du renouvellement de la population, on peut dans ce cas retenir le meilleur des individus choisi parmi les parents et les enfants, individu qui sera alors dans la nouvelle population.

Cette approche augmente le risque de convergence prématurée car elle est très élitiste (élitisme fort).

Parallélisation des problèmes mono-objectif

Les approches proposées ici sont compatibles avec les cartes General Purpose Graphic Processing Unit (GPGPU) et un parallélisme SIMD (Single Instruction Multiple Data) des cartes graphiques) ce qui permet de mettre en œuvre un parallélisme très fort en fonction

du nombre de cœurs hébergés par la carte.

Parallélisme à l'identique Une approche simple de parallélisme consiste simplement à paralléliser à l'identique un AE dans lequel on fait un remplacement générationnel sans élitisme. Il suffit de multiplier le nombre de processeurs et de subdiviser la population par le nombre de processeurs. Un élitisme (fort) peut être introduit au niveau de la sélection des enfants comme nous l'avons expliqué précédemment.

Réduction générationnelle sans remise Au niveau du renouvellement de la population avec élitisme, on ne peut pas appliquer un opérateur de sélection par tournoi (qui est parallélisable) car on prendrait le risque de retrouver plusieurs fois le même individu dans la population, car le tirage est avec remise. L'idée ici pour assurer le parallélisme et l'élitisme au niveau du renouvellement de la population est de subdiviser la population globale des parents et des enfants en sous ensembles distincts et d'effectuer une opération élitiste de renouvellement sur chaque sous-ensemble. Cela permet de ne pas sélectionner plusieurs fois les mêmes individus et de paralléliser.

Parallélisation de l'évaluation seule L'évaluation dans un AE est souvent très coûteuse en calcul, une manière de faire consiste dans ce cas à paralléliser l'étape d'évaluation seule. Le gain peut être très important bien que borné par les étapes non parallélisables (loi d'Amdhal qui dit que la partie séquentielle d'un algorithme est celle qui limite le parallélisme).

Parallélisation des problèmes multi-objectifs

Dans les AE multi-critères le problème est que la détermination du rang d'un individu (ranking) et donc la stratégie élitiste passe par une comparaison des individus entre eux. Ce n'est évidemment pas parallélisable.

La solution passe par l'introduction de mécanismes stochastiques qui effectuent une comparaison non déterministe sur un sous ensemble d'individus tirés au hasard. L'idée générale est qu'un certain "flou" aléatoire du ranking n'est pas préjudiciable au bon fonctionnement d'un AE, au contraire il est même nécessaire pour éviter une convergence prématurée.

Archive based Stochastic Random Evolutionnary Algorithm

[SHARMA et COLLET, 2010] présentent un exemple de parallélisme sur un algorithme multi-objectifs. De bons individus initiaux sont stockés dans une archive et les individus de la population sont dotés d'un rang approximatif qui permet de les comparer au contenu de l'archive et de renouveler l'archive.

L'archive de petite taille permet de comparer très vite le rang des individus à ceux de l'archive et de réduire la complexité. La dominance comporte en fait dans ce cas une composante aléatoire qui a l'avantage de limiter le risque de convergence prématurée de l'algorithme.

La complexité de l'algorithme est en $O(m \cdot a \cdot n)$, où m est le nombre d'objectifs, a la taille d'une archive attachée à chaque cœur de calcul et n le nombre d'individus.

Pour éviter de renouveler les doublons dus à cette comparaison locale, le tirage des parents est purement aléatoire. Les performances obtenues sont remarquables en terme

de temps de calcul et résolution du front de Pareto pour des problèmes types divers et permettent de travailler sur des populations de très grandes tailles, jusqu'à 1M d'individus, avec des gains en temps de calculs d'autant plus grands par rapport à NSGA-II lorsque la population est grande.

Parallélisme en îlots L'idée du parallélisme en îlots est de subdiviser la population en sous populations qui sont traitées de manière parallèle et indépendante.

Pour maintenir une diversité dans la sous population et assurer une convergence globale, on introduit des échanges d'individus dans les différentes sous populations comme cela se produisait pour l'observation de Darwin sur des espèces de pinsons sur différentes îles des Galapagos [LAMICHHANEY et collab., 2015].

Quelques individus échangés peuvent suffire à maintenir la diversité nécessaire et à assurer la convergence de l'algorithme vers un optimum global. Cette approche simple en terme de parallélisme a été testée sur des fonctions objectives aux caractéristiques très complexes et a permis d'obtenir de très bons résultats en particulier en terme de temps de calcul [GONG et FUKUNAGA, 2011].

Elle peut être mise en œuvre assez facilement puisqu'il s'agit simplement de subdiviser la population et d'organiser des échanges entre les sous populations sur des machines multi-cœurs ou dans un contexte distribué.

2.4.6 Justification mathématique des algorithmes évolutionnaires

Comme pour la méta-heuristique de simulated annealing, il existe des travaux de justification mathématique des AE. Ces tentatives peuvent être plus ou moins abouties. L'efficacité pratique de ces algorithmes a paradoxalement nuit à leur théorisation, cependant il existe aujourd'hui de nombreux travaux.

Tout réside dans les hypothèses de travail permettant d'illustrer telle ou telle propriété de ces algorithmes. Ces preuves restent d'un intérêt limité puisque la durée de convergence peut-être infinie.

Algorithmes mono-objectifs

Avec les données suivantes on a pour les algorithmes mono-objectifs:

- x une variable de décision pour un problème d'optimisation mono-objectif
- I l'espace des valeurs réalisables de x
- F une fonction de fitness (problème mono-objectif appliquée à x)
- t le numéro de la génération et $\mathrm{P}(t)$ la population de la génération t

[BACK, 1996] prouve qu'avec ces hypothèses, un AE mono-objectif converge avec une probabilité 1 sous les conditions suivantes:

$\forall x, x' \in \mathrm{I}$, x' est atteignable depuis x par le moyen des opérateurs de croisement et de mutation. De plus, la séquence de population $\mathrm{P}(0), \mathrm{P}(1) \ldots \mathrm{P}(n)$ est monotone, c'est-à-dire que l'on a un ordre total sur les valeurs min de la fonction objectif dans la population d'une génération à l'autre:

$$\forall t : \min \mathrm{F}(x(t+1) | x(t+1) \in \mathrm{P}(t+1)) \leq \min \mathrm{F}(x(t) | x(t) \in \mathrm{P}(t)) \tag{2.4}$$

Cette définition de la monotonie définit un ordre total sur les valeurs de la fonction objective et n'est donc pas adaptée aux problèmes multi-objectifs dans lesquels la relation d'ordre sur les valeurs des fitness est partielle (définition du front de Pareto). Des développements théoriques complémentaires permettent cependant de démontrer un théorème similaire dans le contexte multi-objectifs.

Algorithmes multi-objectifs

Dans le contexte multi-objectifs des arguments similaires ont été développés par [COELLO et collab., 2006].

Si l'on considère un algorithme évolutionnaire multi-objectifs à population, la condition 2.4.6 permet de définir une notion de convergence vers le front de Pareto qui est solution du problème. On pose:

- $PF_{current}$: le front de Pareto courant établi au fil des itérations de l'algorithme,
- PF_{known}: le front de Pareto calculé sur la population courante,
- PF_{true}: le front de Pareto, solution du problème,
- $M(x)$ minimum de l'ensemble x au sens de la relation d'ordre partiel permettant de définir un front de Pareto,
- $\preceq$ la relation d'ordre partiel qui permet de définir le front de Pareto

Si l'on a la même condition que dans le cas mono-objectif, c'est à dire : $\forall x, x' \in I$, x' est atteignable depuis x par le moyen des opérateurs de croisement et de mutation.

et si l'on a la condition 2.5:

$$PF_{known}(t+1) = M(PF_{current}(t) \cup PF_{known}, \preceq) \tag{2.5}$$

Alors un algorithme multi-objectif converge avec une probabilité $p = 1$ vers le front de Pareto optimal noté P_{true} et on a:

$$Prob[\lim_{t\to\infty}(P_{true} = P_{known}(t))] = 1 \tag{2.6}$$

Dans le même ordre d'idée [RUDOLPH, 1998] définit un algorithme multi-objectif à population dont il démontre la convergence vers l'ensemble des éléments minimaux (au sens d'un ordre partiel nécessaire à la définition d'un front de Pareto), c'est à dire le front de Pareto avec une probabilité de 1 dans un espace de recherche fini.

Deux propriétés sont nécessaires à l'algorithme pour garantir cette convergence:

- Un noyau de variation positif (A positive variation kernel) (c'est-à-dire que les opérations de mutations et de croisement assurent avec une certaine probabilité la production de meilleurs individus et le parcours de l'espace des solutions.
- Une stratégie de préservation fortement élitiste c'est à dire qu'on remplace systématiquement les individus moins bons par de meilleurs dans la population.

En fait les conditions de convergence se résument de la manière suivante dans la structure de l'algorithme: L' AEMO est fondé sur une comparaison systématique des individus des populations de parents et de descendants.

Les solutions non dominantes de la population des descendants sont comparées avec celles des parents pour construire l'ensemble global des solutions non dominées qui devient la population de parents de la prochaine itération. Si la taille de cet ensemble n'est pas supérieure à la taille de la population fixée, d'autres individus de la population sont inclus dans la population courante pour atteindre la taille fixée.

Conclusion sur les AE

Dans cet état de l'art, nous avons étudié les potentialités des AE comme outil de résolution de problèmes. Ces algorithmes miment l'évolution du vivant pour rechercher les solutions les meilleures dans l'espace des états du problème. Ils permettent une très grande diversité d'approches, et leur flexibilité permet de les adapter à une très large catégorie de problèmes tout en recherchant des méthodes génériques. Ils permettent d'innover en fonction des problèmes à traiter malgré la production foisonnante dont ils ont fait l'objet. C'est pourquoi nous les avons retenus pour aborder le problème du placement de services réseaux.

Bibliography

2017, «Sdn control plane performance an onos™ project white paper», URL http://onosproject.org/wp-content/uploads/2017/08/ONOS_Performance_White_Paper-2.pdf.

A. DIAZ-GOMEZ, P. et D. HOUGEN. 2007, «Initial population for genetic algorithms: A metric approach.», .

ADDIS, B., D. BELABED, M. BOUET et S. SECCI. 2015, «Virtual network functions placement and routing optimization», dans *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, p. 171–177, doi:10.1109/CloudNet.2015.7335301.

AUGER, A. et N. HANSEN. 2005, «A restart cma evolution strategy with increasing population size», p. 1769–1776, doi:10.1109/CEC.2005.1554902.

AYKUT OZSOY, F. et M. C. PINAR. 2006, «An exact algorithm for the capacitated vertex p-center problem», *Comput. Oper. Res.*, vol. 33, nº 5, doi:10.1016/j.cor.2004.09.035, p. 1420–1436, ISSN 0305-0548. URL http://dx.doi.org/10.1016/j.cor.2004.09.035.

AYOUB, J. N., W. H. SAAFIN et B. Z. KAHHALEH. 2000, «k-terminal reliability of communication networks», dans *ICECS 2000. 7th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.00EX445)*, vol. 1, p. 374–377 vol.1, doi:10.1109/ICECS.2000.911559.

BÄCK, T. 1993, «Optimal mutation rates in genetic search», dans *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN 1-55860-299-2, p. 2–8. URL http://dl.acm.org/citation.cfm?id=645513.657408.

BACK, T. 1996, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, ISBN 9780195356700. URL https://books.google.fr/books?id=htJHI1UrL7IC.

BAR-ILAN, J. et D. PELEG. 1991, «Approximation algorithms for selecting network centers», dans *Algorithms and Data Structures*, édité par F. Dehne, J.-R. Sack et N. Santoro, Springer Berlin Heidelberg, Berlin, Heidelberg, p. 343–354.

BARACH, D., L. LINGUAGLOSSA, D. MARION, P. PFISTER, S. PONTARELLI et D. ROSSI. 2018, «High-speed software data plane via vectorized packet processing», *IEEE Communications Magazine*, vol. 56, nº 12, doi:10.1109/MCOM.2018.1800069, p. 97–103, ISSN 0163-6804.

BARBETTE, T., C. SOLDANI et L. MATHY. 2015, «Fast userspace packet processing», dans *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '15, IEEE Computer Society, Washington, DC, USA, ISBN 978-1-4673-6632-8, p. 5–16. URL http://dl.acm.org/citation.cfm?id=2772722.2772727.

BENSON, T., A. AKELLA et D. A. MALTZ. 2010, «Network traffic characteristics of data centers in the wild», dans *Proceedings of the 10th ACM SIGCOMM Conference on*

Internet Measurement, IMC '10, ACM, New York, NY, USA, ISBN 978-1-4503-0483-2, p. 267–280, doi:10.1145/1879141.1879175. URL http://doi.acm.org/10.1145/1879141.1879175.

BERMAN, M., C. ELLIOTT et L. LANDWEBER. 2014, «Geni: Large-scale distributed infrastructure for networking and distributed systems research», dans *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, p. 156–161, doi:10.1109/CCE.2014.6916696.

BIANCHI, G., M. BONOLA, A. CAPONE et C. CASCONE. 2014, «Openstate: Programming platform-independent stateful openflow applications inside the switch», *SIGCOMM Comput. Commun. Rev.*, vol. 44, nº 2, p. 44–51, ISSN 0146-4833.

BIANCO, A., P. GIACCONE, S. D. DOMENICO et T. ZHANG. 2016, «The role of inter-controller traffic for placement of distributed SDN controllers», *CoRR*, vol. abs/1605.09268. URL http://arxiv.org/abs/1605.09268.

BLICKLE, T. et L. THIELE. 1995, «A comparison of selection schemes used in genetic algorithms», cahier de recherche, GLORIASTRASSE 35, CH-8092 ZURICH: SWISS FEDERAL INSTITUTE OF TECHNOLOGY (ETH) ZURICH, COMPUTER ENGINEERING AND COMMUNICATIONS NETWORKS LAB (TIK.

BLUM, C. et A. ROLI. 2003, «Metaheuristics in combinatorial optimization: Overview and conceptual comparison», *ACM Comput. Surv.*, vol. 35, nº 3, doi:10.1145/937503.937505, p. 268–308, ISSN 0360-0300. URL http://doi.acm.org/10.1145/937503.937505.

BO, H., W. YOUKE, W. CHUAN'AN et W. YING. 2016, «The controller placement problem for software-defined networks», dans *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, p. 2435–2439, doi:10.1109/CompComm.2016.7925136.

BOSSHART, P., D. DALY, G. GIBB, M. IZZARD, N. MCKEOWN, J. REXFORD, C. SCHLESINGER, D. TALAYCO, A. VAHDAT, G. VARGHESE et D. WALKER. 2014, «P4: Programming protocol-independent packet processors», *SIGCOMM Comput. Commun. Rev.*, vol. 44, nº 3, p. 87–95, ISSN 0146-4833.

BRANDES, U. 2001, «A faster algorithm for betweenness centrality», *Journal of Mathematical Sociology*, vol. 25, p. 163–177.

CARPIO, F., S. DHAHRI et A. JUKAN. 2017, «VNF placement with replication for load balancing in NFV networks», dans *Proc. of IEEE ICC*, p. 1–6, doi:10.1109/ICC.2017.7996515.

CHATRAS, B. 2015, «Virtualisation réseau nfv: Cadre architectural et principes techniques», *ITN school*.

CHOI, S., X. LONG, M. SHAHBAZ, S. BOOTH, A. KEEP, J. MARSHALL et C. KIM. 2017, «Pvpp: A programmable vector packet processor», dans *Proceedings of the Symposium on SDN Research*, SOSR '17, ACM, New York, NY, USA, ISBN 978-1-4503-4947-5, p. 197–198.

COELLO, C. A. C., G. B. LAMONT et D. A. V. VELDHUIZEN. 2006, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, Springer-Verlag, Berlin, Heidelberg, ISBN 0387332545.

DA CRUZ MARCUZZO, L., V. F. GARCIA, V. CUNHA, D. CORUJO, J. P. BARRACA, R. L. AGUIAR, A. E. SCHAEFFER-FILHO, L. Z. GRANVILLE et C. R. P. DOS SANTOS. 2017, «Click-on-osv: A platform for running click-based middleboxes», dans *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, p. 885–886, doi:10.23919/INM.2017.7987396.

CURTIS, A. R., J. C. MOGUL, J. TOURRILHES, P. YALAG, P. SHARMA et S. BANERJEE. 2011, «Devoflow: Scaling flow management for high-performance networks», dans *In ACM SIGCOMM*.

DAHAL, K. P. et J. MCDONALD. 1997, «Generational and steady state genetic algorithms for generator maintenance scheduling problems», URL `http://hdl.handle.net/10454/2453`, the aim of generator maintenance scheduling(GMS) in an electric power system is to allocate a propermaintenance timetable for generators while maintaining a high-system reliability, reducing total production cost, extendinggenerator life time etc. In order to solve this complex problema genetic algorithm technique is proposed here. The paperdiscusses the implementation of GAs to GMS problems withtwo approaches: generational and steady state. The results ofapplying these GAs to a test GMS problem based on apractical power system scenario are presented and analysed.The effect of different GA parameters is also studied.

DAI, W., Z. GU, X. LIN, Q. HUA et F. C. M. LAU. 2015, «Minimum control latency of dynamic networks», dans *2015 IEEE Conference on Computer Communications (INFOCOM)*, ISSN 0743-166X, p. 648–656, doi:10.1109/INFOCOM.2015.7218433.

DEB, K., A. PRATAP, S. AGARWAL et T. MEYARIVAN. 2002a, «A fast and elitist multiobjective genetic algorithm: Nsga-ii», *IEEE Transactions on Evolutionary Computation*, vol. 6, n° 2, doi:10.1109/4235.996017, p. 182–197, ISSN 1089-778X.

DEB, K., A. PRATAP, S. AGARWAL et T. MEYARIVAN. 2002b, «A fast and elitist multiobjective genetic algorithm: Nsga-ii», *Trans. Evol. Comp*, vol. 6, n° 2, doi:10.1109/4235.996017, p. 182–197, ISSN 1089-778X. URL `http://dx.doi.org/10.1109/4235.996017`.

DIXIT, A., F. HAO, S. MUKHERJEE, T. LAKSHMAN et R. KOMPELLA. 2013, «Towards an elastic distributed sdn controller», dans *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, ACM, New York, NY, USA, ISBN 978-1-4503-2178-5, p. 7–12, doi:10.1145/2491185.2491193. URL `http://doi.acm.org/10.1145/2491185.2491193`.

DIXIT, A., F. HAO, S. MUKHERJEE, T. V. LAKSHMAN et R. R. KOMPELLA. 2014, «Elasticon; an elastic distributed sdn controller», dans *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, p. 17–27.

DOBBELAERE, P. et K. SHEYKH ESMAILI. 2017, «Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry paper», p. 227–238, doi:10.1145/3093742.3093908.

DWARAKI, A. et T. WOLF. 2016, «Adaptive service-chain routing for virtual network functions in software-defined networks», dans *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, HotMIddlebox '16, ACM, New York, NY, USA, ISBN 978-1-4503-4424-1, p. 32–37, doi:2940147.2940148. URL `http://doi.acm.org/2940147.2940148`.

EPPSTEIN, D. 1999, «Finding the k shortest paths», *SIAM J. Comput.*, vol. 28, nº 2, doi: 10.1137/S0097539795290477, p. 652–673, ISSN 0097-5397. URL http://dx.doi.org/10.1137/S0097539795290477.

ETSI. 2014, «Network Functions Virtualisation (NFV); Management and Orchestration», Group Specification (GS) NFV-MAN 001, European Telecommunication Standard Institute (ETSI). Version 1.1.1.

FALKENAUER, E. 1993, «The grouping genetic algorithms: Widening the scope of the gas», *Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL)*, vol. 33.

FARREL, A., J.-P. VASSEUR et J. ASH. 2006, «A path computation element (PCE)-based architecture», cahier de recherche.

FEAMSTER, N., J. BORKENHAGEN et J. REXFORD. 2003, «Guidelines for interdomain traffic engineering», *SIGCOMM Comput Commu Rev*, vol. 33, nº 5.

FELDMAN, V. 2008, «Evolvability from learning algorithms», dans *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, ACM, New York, NY, USA, ISBN 978-1-60558-047-0, p. 619–628, doi:10.1145/1374376.1374465. URL http://doi.acm.org/10.1145/1374376.1374465.

FISHER, R. A. et J. H. BENNETT. 1999, *The genetical theory of natural selection : a complete variorum edition / by R.A. Fisher ; edited with a foreword and notes by J.H. Bennett*, Oxford University Press Oxford, ISBN 0198504403, xxii, xiv, 318 p., [4] leaves of plates : p..

FU, Y., J. BI, K. GAO, Z. CHEN, J. WU et B. HAO. 2014, «Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks», dans *2014 IEEE 22nd International Conference on Network Protocols*, ISSN 1092-1648, p. 569–576, doi: 10.1109/ICNP.2014.91.

GILBERT, S. et N. LYNCH. 2012, «Perspectives on the cap theorem», *Computer*, vol. 45, nº 2, doi:10.1109/MC.2011.389, p. 30–36, ISSN 0018-9162. URL https://doi.org/10.1109/MC.2011.389.

GOLDBERG, D. E. 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1re éd., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, ISBN 0201157675.

GONG, Y. et A. FUKUNAGA. 2011, «Distributed island-model genetic algorithms using heterogeneous parameter settings», dans *2011 IEEE Congress of Evolutionary Computation (CEC)*, ISSN 1941-0026, p. 820–827, doi:10.1109/CEC.2011.5949703.

GUAN, X., B. CHOI et S. SONG. 2013, «Reliability and scalability issues in software defined network frameworks», dans *2013 Second GENI Research and Educational Experiment Workshop*, p. 102–103, doi:10.1109/GREE.2013.28.

GUO, M. et P. BHATTACHARYA. 2013, «Controller placement for improving resilience of software-defined networks», dans *2013 Fourth International Conference on Networking and Distributed Computing*, ISSN 2165-4999, p. 23–27, doi:10.1109/ICNDC.2013.15.

HAN, L., Z. LI, W. LIU, K. DAI et W. QU. 2016, «Minimum control latency of sdn controller placement», dans *2016 IEEE Trustcom/BigDataSE/ISPA*, ISSN 2324-9013, p. 2175–2180, doi:10.1109/TrustCom.2016.0334.

HASSAS YEGANEH, S. et Y. GANJALI. 2012, «Kandoo: A framework for efficient and scalable offloading of control applications», dans *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, ACM, New York, NY, USA, ISBN 978-1-4503-1477-0, p. 19–24, doi:10.1145/2342441.2342446. URL http://doi.acm.org/10.1145/2342441.2342446.

HELLER, B., R. SHERWOOD et N. MCKEOWN. 2012, «The controller placement problem», dans *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, ACM, New York, NY, USA, ISBN 978-1-4503-1477-0, p. 7–12, doi:10.1145/2342441.2342444. URL http://doi.acm.org/10.1145/2342441.2342444.

HOCHBAUM, D. S. et D. B. SHMOYS. 1984, «Powers of graphs: A powerful approximation technique for bottleneck problems», dans *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, ACM, New York, NY, USA, ISBN 0-89791-133-4, p. 324–333, doi:10.1145/800057.808697. URL http://doi.acm.org/10.1145/800057.808697.

HOCK, D., M. HARTMANN, S. GEBERT, M. JARSCHEL, T. ZINNER et P. TRAN-GIA. 2013, «Pareto-optimal resilient controller placement in sdn-based core networks», dans *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, p. 1–9, doi:10.1109/ITC.2013.6662939.

HOCK, D., M. HARTMANN, S. GEBERT, T. ZINNER et P. TRAN-GIA. 2014, «Poco-plc: Enabling dynamic pareto-optimal resilient controller placement in sdn networks», dans *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, p. 115–116, doi:10.1109/INFCOMW.2014.6849182.

HOLZINGER, A., D. BLANCHARD, M. BLOICE, K. HOLZINGER, V. PALADE et R. RABADAN. 2014, «Darwin, lamarck, or baldwin: Applying evolutionary algorithms to machine learning techniques», dans *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 02*, WI-IAT '14, IEEE Computer Society, Washington, DC, USA, ISBN 978-1-4799-4143-8, p. 449–453, doi:10.1109/WI-IAT.2014.132. URL http://dx.doi.org/10.1109/WI-IAT.2014.132.

HU, Y., T. LUO, W. WANG et C. DENG. 2016, «On the load balanced controller placement problem in software defined networks», dans *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, p. 2430–2434, doi:10.1109/CompComm.2016.7925135.

HU, Y., W. WENDONG, X. GONG, X. QUE et C. SHIDUAN. 2013, «Reliability-aware controller placement for software-defined networks», dans *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, ISSN 1573-0077, p. 672–675.

INTEL. 2017, «Fd.io - vector packet processing white paper», cahier de recherche, INTEL.

JAIN, S., A. KUMAR, S. MANDAL, J. ONG, L. POUTIEVSKI, A. SINGH, S. VENKATA, J. WANDERER, J. ZHOU, M. ZHU, J. ZOLLA, U. HÖLZLE, S. STUART et A. VAHDAT. 2013, «B4:

Experience with a globally-deployed software defined wan», *SIGCOMM Comput. Commun. Rev.*, vol. 43, nº 4.

JAKMA, P. et D. LAMPARTER. 2014, «Introduction to the quagga routing suite», *Network, IEEE*, vol. 28, doi:10.1109/MNET.2014.6786612, p. 42–48.

JALILI, A., V. AHMADI, M. KESHTGARI et M. KAZEMI. 2015, «Controller placement in software-defined wan using multi objective genetic algorithm», *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, p. 656–662.

JAYASENA, K. P. N., L. LI, M. ABD ELAZIZ et S. XIONG. 2018, «Multi-objective energy efficient resource allocation using virus colony search (vcs) algorithm», dans *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, p. 766–773, doi:10.1109/HPCC/SmartCity/DSS.2018.00130.

JIMÉNEZ, Y., C. CERVELLÓ-PASTOR et A. J. GARCÍA. 2014, «On the controller placement for designing a distributed sdn control layer», dans *2014 IFIP Networking Conference*, p. 1–9, doi:10.1109/IFIPNetworking.2014.6857117.

JIN, X., L. E. LI, L. VANBEVER et J. REXFORD. 2013, «Softcell: Scalable and flexible cellular core network architecture», dans *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, ACM, New York, NY, USA, ISBN 978-1-4503-2101-3, p. 163–174, doi:10.1145/2535372.2535377. URL `http://doi.acm.org/10.1145/2535372.2535377`.

JOUET, S. et D. P. PEZAROS. 2017, «Bpfabric: Data plane programmability for software defined networks», dans *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, p. 38–48.

KANADE, V. 2011, «Evolution with recombination», dans *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, ISSN 0272-5428, p. 837–846, doi:10.1109/FOCS.2011.24.

KHULLER, S. et Y. J. SUSSMANN. 1996, «The capacitated k-center problem», dans *In Proceedings of the 4th Annual European Symposium on Algorithms, Lecture Notes in Computer Science 1136*, Springer, p. 152–166.

KILLI, B. P. R. et S. V. RAO. 2017, «Capacitated next controller placement in software defined networks», *IEEE Transactions on Network and Service Management*, vol. 14, nº 3, doi:10.1109/TNSM.2017.2720699, p. 514–527, ISSN 1932-4537.

KILLI, B. P. R., E. A. REDDY et S. V. RAO. 2018, «Cooperative game theory based network partitioning for controller placement in SDN», dans *10th International Conference on Communication Systems & Networks, COMSNETS 2018, Bengaluru, India, January 3-7, 2018*, p. 105–112, doi:10.1109/COMSNETS.2018.8328186. URL `https://doi.org/10.1109/COMSNETS.2018.8328186`.

KIM, J., S. HUH, K. JANG, K. PARK et S. MOON. 2012, «The power of batching in the click modular router», dans *Proceedings of the Asia-Pacific Workshop on Systems*, APSYS '12, ACM, New York, NY, USA, ISBN 978-1-4503-1669-9, p. 14:1–14:6, doi:10.1145/2349896.2349910. URL `http://doi.acm.org/10.1145/2349896.2349910`.

KNIGHT, S., H. X. NGUYEN, N. FALKNER, R. A. BOWDEN et M. ROUGHAN. 2011, «The internet topology zoo.», *IEEE Journal on Selected Areas in Communications*, vol. 29, nº 9, p. 1765–1775. URL http://dblp.uni-trier.de/db/journals/jsac/jsac29.html#KnightNFBR11.

KOHLER, E., R. MORRIS, B. CHEN, J. JANNOTTI et M. F. KAASHOEK. 2000, «The click modular router», *ACM Trans. Comput. Syst.*, vol. 18, nº 3, doi:10.1145/354871.354874, p. 263–297, ISSN 0734-2071. URL http://doi.acm.org/10.1145/354871.354874.

KOPONEN, T., M. CASADO, N. GUDE, J. STRIBLING, L. POUTIEVSKI, M. ZHU, R. RAMANATHAN, Y. IWATA, H. INOUE, T. HAMA et S. SHENKER. 2010, «Onix: A distributed control platform for large-scale production networks», dans *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, USENIX Association, Berkeley, CA, USA, p. 351–364. URL http://dl.acm.org/citation.cfm?id=1924943.1924968.

KORKMAZ, T. et M. KRUNZ. 2001, «Multi-constrained optimal path selection», dans *IEEE INFOCOM*, vol. 2, ISSN 0743-166X, p. 834–843, doi:10.1109/INFCOM.2001.916274.

KOZA, J. R. 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, USA, ISBN 0-262-11170-5.

KSENTINI, A., M. BAGAA, T. TALEB et I. BALASINGHAM. 2016, «On using bargaining game for optimal placement of sdn controllers», dans *2016 IEEE International Conference on Communications (ICC)*, ISSN 1938-1883, p. 1–6, doi:10.1109/ICC.2016.7511136.

KUIPERS, F., T. KORKMAZ, M. KRUNZ et P. V. MIEGHEM. 2004, «Performance evaluation of constraint-based path selection algorithms», *IEEE Network*, vol. 18, nº 5, doi:10.1109/MNET.2004.1337731, p. 16–23, ISSN 0890-8044.

KUIPERS, F., P. V. MIEGHEM, T. KORKMAZ et M. KRUNZ. 2002, «An overview of constraint-based path selection algorithms for QoS routing», *IEEE Communications Magazine*, vol. 40, nº 12, doi:10.1109/MCOM.2002.1106159, p. 50–55, ISSN 0163-6804.

L. MILLER, B. et D. E. GOLDBERG. 1995, «Genetic algorithms, tournament selection, and the effects of noise», *Complex Systems*, vol. 9.

LAMICHHANEY, S., J. BERGLUND, M. SÄLLMAN ALMÉN, K. MAQBOOL, M. GRABHERR, A. MARTINEZ BARRIO, M. PROMEROVÁ, C.-J. RUBIN, C. WANG, N. ZAMANI, B. ROSEMARY GRANT, P. R GRANT, M. WEBSTER et L. ANDERSSON. 2015, «Evolution of darwin's finches and their beaks revealed by genome sequencing», *Nature*, vol. 518, doi:10.1038/nature14181.

LANGE, S., S. GEBERT, J. SPOERHASE, P. RYGIELSKI, T. ZINNER, S. KOUNEV et P. TRAN-GIA. 2015a, «Specialized heuristics for the controller placement problem in large scale sdn networks», dans *2015 27th International Teletraffic Congress*, p. 210–218, doi:10.1109/ITC.2015.32.

LANGE, S., S. GEBERT, T. ZINNER, P. TRAN-GIA, D. HOCK, M. JARSCHEL et M. HOFFMANN. 2015b, «Heuristic approaches to the controller placement problem in large scale sdn networks», *IEEE Transactions on Network and Service Management*, vol. 12, nº 1, doi:10.1109/TNSM.2015.2402432, p. 4–17, ISSN 1932-4537.

LANGE, S., A. GRIGORJEW, T. ZINNER, P. TRAN-GIA et M. JARSCHEL. 2017, «A multi-objective heuristic for the optimization of virtual network function chain placement», dans *2017 29th International Teletraffic Congress (ITC 29)*, vol. 1, p. 152–160, doi:10.23919/ITC.2017.8064351.

LAUFER, R., M. GALLO, D. PERINO et A. NANDUGUDI. 2016, «Climb: Enabling network function composition with click middleboxes», *SIGCOMM Comput. Commun. Rev.*, vol. 46, nº 4, doi:10.1145/3027947.3027951, p. 17–22, ISSN 0146-4833. URL http://doi.acm.org/10.1145/3027947.3027951.

LEE, W. C., M. G. HLUCHYI et P. A. HUMBLET. 1995, «Routing subject to quality of service constraints in integrated communication networks», *IEEE Network*, vol. 9, nº 4, doi:10.1109/65.397043, p. 46–55, ISSN 0890-8044.

LEVIN, D., A. WUNDSAM, B. HELLER, N. HANDIGOL et A. FELDMANN. 2012, «Logically centralized?: State distribution trade-offs in software defined networks», dans *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, ACM, New York, NY, USA, ISBN 978-1-4503-1477-0, p. 1–6, doi:10.1145/2342441.2342443. URL http://doi.acm.org/10.1145/2342441.2342443.

LI, B., K. TAN, L. L. LUO, Y. PENG, R. LUO, N. XU, Y. XIONG, P. CHENG et E. CHEN. 2016, «Clicknp: Highly flexible and high performance network processing with reconfigurable hardware», dans *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, ACM, New York, NY, USA, ISBN 978-1-4503-4193-6, p. 1–14, doi:10.1145/2934872.2934897. URL http://doi.acm.org/10.1145/2934872.2934897.

LIAO, J., H. SUN, J. WANG, Q. QI, K. LI et T. LI. 2017, «Density cluster based approach for controller placement problem in large-scale software defined networkings», *Comput. Netw.*, vol. 112, nº C, doi:10.1016/j.comnet.2016.10.014, p. 24–35, ISSN 1389-1286. URL https://doi.org/10.1016/j.comnet.2016.10.014.

LIU, G. et K. G. RAMAKRISHNAN. 2001, «A*prune: an algorithm for finding k shortest paths subject to multiple constraints», dans *IEEE INFOCOM*, vol. 2, ISSN 0743-166X, p. 743–749, doi:10.1109/INFCOM.2001.916263.

LIU, Y.-Y., J.-J. SLOTINE et A.-L. BARABASI. 2011, «Controllability of complex networks», vol. 473, p. 167–73.

LOVÁSZ, L. 1975, «On the ratio of optimal integral and fractional covers», *Discrete Math.*, vol. 13, nº 4, doi:10.1016/0012-365X(75)90058-8, p. 383–390, ISSN 0012-365X. URL http://dx.doi.org/10.1016/0012-365X(75)90058-8.

MANN, Z. A. 2015, «Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms», *ACM Comput. Surv.*, vol. 48, nº 1, doi:10.1145/2797211, p. 11:1–11:34, ISSN 0360-0300. URL http://doi.acm.org/10.1145/2797211.

MARTINS, J., M. AHMED, C. RAICIU, V. OLTEANU, M. HONDA, R. BIFULCO et F. HUICI. 2014, «Clickos and the art of network function virtualization», dans *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, USENIX Association, Berkeley, CA, USA, ISBN 978-1-931971-09-6, p. 459–473. URL http://dl.acm.org/citation.cfm?id=2616448.2616491.

MIEGHEM, P. V., H. D. NEVE et F. KUIPERS. 2001, «Hop-by-hop quality of service routing», *Computer Networks*, vol. 37, nº 3, doi:https://doi.org/10.1016/S1389-1286(01)00222-5, p. 407 – 423, ISSN 1389-1286.

MITCHELL, G. G., D. BARNES et M. MCCARVILLE. «Generepair- a repair operator for genetic algorithms», .

MUQADDAS, A. S., A. BIANCO, P. GIACCONE et G. MAIER. 2016, «Inter-controller traffic in onos clusters for sdn networks», dans *2016 IEEE International Conference on Communications (ICC)*, p. 1–6, doi:10.1109/ICC.2016.7511034.

NAGATA, Y. et S. KOBAYASHI. 1999, «An analysis of edge assembly crossover for the traveling salesman problem», dans *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, vol. 3, ISSN 1062-922X, p. 628–633 vol.3, doi:10.1109/ICSMC.1999.823285.

NEVE, H. D. et P. V. MIEGHEM. 2000, «TAMCRA: a tunable accuracy multiple constraints routing algorithm», *Computer Communications*, vol. 23, nº 7, doi:https://doi.org/10.1016/S0140-3664(99)00225-X, p. 667 – 679, ISSN 0140-3664.

NOWAK, M. 2006, *Evolutionary Dynamics*, Harvard University Press, ISBN 9780674023383. URL `https://books.google.fr/books?id=YXrIRDuAbE0C`.

NOWAK, M. 2007, *Evolutionary Dynamics: Exploring the Equations of Life*, vol. 82.

OBADIA, M., M. BOUET, J. ROUGIER et L. IANNONE. 2015, «A greedy approach for minimizing sdn control overhead», dans *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, p. 1–5, doi:10.1109/NETSOFT.2015.7116135.

OSMAN, M. S., M. A. ABO-SINNA et A. A. MOUSA. 2006, «Epsilon-dominance based multiobjective genetic algorithm for economic emission load dispatch optimization problem», dans *2006 Eleventh International Middle East Power Systems Conference*, vol. 2, p. 576–581.

PALIWAL, M., D. SHRIMANKAR et O. TEMBHURNE. 2018, «Controllers in sdn: A review report», *IEEE Access*, vol. 6, doi:10.1109/ACCESS.2018.2846236, p. 36 256–36 270, ISSN 2169-3536.

PANDA, A., C. SCOTT, A. GHODSI, T. KOPONEN et S. SHENKER. 2013a, «Cap for networks», HotSDN '13. URL `http://www.eecs.berkeley.edu/~rcs/research/cap-paper.pdf`.

PANDA, A., C. SCOTT, A. GHODSI, T. KOPONEN et S. SHENKER. 2013b, «Cap for networks», dans *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, ACM, New York, NY, USA, ISBN 978-1-4503-2178-5, p. 91–96, doi:10.1145/2491185.2491186. URL `http://doi.acm.org/10.1145/2491185.2491186`.

PETROWSKI, A. et S. BEN HAMIDA. 2017, *Evolutionary Algorithms*, John Wiley & Sons, Ltd. URL `https://hal.archives-ouvertes.fr/hal-02091413`.

PFAFF, B., J. PETTIT, T. KOPONEN, E. J. JACKSON, A. ZHOU, J. RAJAHALME, J. GROSS, A. WANG, J. STRINGER, P. SHELAR, K. AMIDON et M. CASADO. 2015, «The design

and implementation of open vswitch», dans *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, USENIX Association, Berkeley, CA, USA, ISBN 978-1-931971-218, p. 117–130. URL http://dl.acm.org/citation.cfm?id=2789770.2789779.

PHEMIUS, K., M. BOUET et J. LEGUAY. 2014, «Disco: Distributed multi-domain sdn controllers», dans *2014 IEEE Network Operations and Management Symposium (NOMS)*, ISSN 1542-1201, p. 1–4, doi:10.1109/NOMS.2014.6838330.

POTVIN, J.-Y. 1996, «Genetic algorithms for the traveling salesman problem», *Annals OR*, vol. 63, p. 337–370.

PRISCO, R. D. et B. LAMPSON. 1997, «Revisiting the paxos algorithm», dans *In Marios Mavronicolas and Philippas Tsigas, editors, Proceedings of the 11th International Workshop on Distributed Algorithms (WDAG 97), volume 1320 of Lecture Notes in Computer Science*, Springer-Verlag, p. 111–125.

PUJOLLE, G. et O. SALVATORI. 2014, *Les réseaux*, Eyrolles, ISBN 9782212138528. URL https://books.google.fr/books?id=R9QnAwAAQBAJ.

RATH, H. K., V. REVOORI, S. M. NADAF et A. SIMHA. 2014, «Optimal controller placement in software defined networks (sdn) using a non-zero-sum game», dans *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, p. 1–6, doi:10.1109/WoWMoM.2014.6918987.

ROS, F. et P. RUIZ. 2014, «Five nines of southbound reliability in software-defined networks», dans *HotSDN 2014 - Proceedings of the ACM SIGCOMM 2014 Workshop on Hot Topics in Software Defined Networking*.

RUDOLPH, G. 1998, «Evolutionary search for minimal elements in partially ordered finite sets», dans *Evolutionary Programming VII*, édité par V. W. Porto, N. Saravanan, D. Waagen et A. E. Eiben, Springer Berlin Heidelberg, Berlin, Heidelberg, p. 345–353.

SALLAHI, A. et M. ST-HILAIRE. 2015, «Optimal model for the controller placement problem in software defined networks», *IEEE Communications Letters*, vol. 19, n° 1, doi:10.1109/LCOMM.2014.2371014, p. 30–33, ISSN 1089-7798.

SANTOS, M. A. S., B. A. A. NUNES, K. OBRACZKA, T. TURLETTI, B. T. DE OLIVEIRA et C. B. MARGI. 2014, «Decentralizing sdn's control plane», dans *39th Annual IEEE Conference on Local Computer Networks*, ISSN 0742-1303, p. 402–405.

SCHAFFER, J. D. 1985, «Multiple objective optimization with vector evaluated genetic algorithms», dans *Proceedings of the 1st International Conference on Genetic Algorithms*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, ISBN 0-8058-0426-9, p. 93–100. URL http://dl.acm.org/citation.cfm?id=645511.657079.

SCHMID, S. et J. SUOMELA. 2013, «Exploiting locality in distributed sdn control», dans *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, ACM, New York, NY, USA, ISBN 978-1-4503-2178-5, p. 121–126, doi:10.1145/2491185.2491198. URL http://doi.acm.org/10.1145/2491185.2491198.

SHAH, N., W. PLISHKER, K. RAVINDRAN et K. KEUTZER. 2004, «Np-click: a productive software development approach for network processors», *IEEE Micro*, vol. 24, nº 5, doi:10.1109/MM.2004.53, p. 45–54, ISSN 0272-1732.

SHAKSHUKI, E., S. GALLAND, A.-U.-H. YASAR, L. ZUCCARO, F. CIMORELLI, F. D. PRISCOLI, C. G. GIORGI, S. MONACO et V. SURACI. 2015, «The 10th international conference on future networks and communications (fnc 2015) / the 12th international conference on mobile systems and pervasive computing (mobispc 2015) affiliated workshops distributed control in virtualized networks», *Procedia Computer Science*, vol. 56, doi:http://dx.doi.org/10.1016/j.procs.2015.07.209, p. 276 – 283, ISSN 1877-0509. URL http://www.sciencedirect.com/science/article/pii/S1877050915016907.

SHARMA, D. et P. COLLET. 2010, «Gpgpu-compatible archive based stochastic ranking evolutionary algorithm (g-asrea) for multi-objective optimization», dans *Parallel Problem Solving from Nature, PPSN XI*, édité par R. Schaefer, C. Cotta, J. Kołodziej et G. Rudolph, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-642-15871-1, p. 111–120.

SHRIVASTWA, A., S. SARAT, K. JACKSON, C. BUNCH, E. SIGLER et T. CAMPBELL. 2016, *OpenStack: Building a Cloud Environment*, Packt Publishing, ISBN 1787123189, 9781787123182.

SIEW MOOI, S. L., A. B. MD SULTAN, M. NASIR SULAIMAN, A. MUSTAPHA et K. Y. LEONG. 2017, «Crossover and mutation operators of genetic algorithms», *International Journal of Machine Learning and Computing*, vol. 7, doi:10.18178/ijmlc.2017.7.1.611, p. 9–12.

SIMPSON, G. G. 1953, «The baldwin effect», *Evolution*, vol. 7, nº 2, doi:10.1111/j.1558-5646.1953.tb00069.x, p. 110–117. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1558-5646.1953.tb00069.x.

HTTPS://WIKI.ONAP.ORG/DISPLAY/ SISUKAPALLI. 2017, «Onap-of project proposal», cahier de recherche.

SUN, W. et R. RICCI. 2013, «Fast and flexible: Parallel packet processing with gpus and click», dans *Architectures for Networking and Communications Systems*, p. 25–35, doi:10.1109/ANCS.2013.6665173.

TOOTOONCHIAN, A. et Y. GANJALI. 2010, «Hyperflow: A distributed control plane for openflow», dans *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, USENIX Association, Berkeley, CA, USA, p. 3–3. URL http://dl.acm.org/citation.cfm?id=1863133.1863136.

TOOTOONCHIAN, A., S. GORBUNOV, Y. GANJALI, M. CASADO et R. SHERWOOD. 2012, «On controller performance in software-defined networks», dans *Proceedings of the 2Nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE'12, USENIX Association, Berkeley, CA, USA, p. 10–10. URL http://dl.acm.org/citation.cfm?id=2228283.2228297.

TUNCER, D., M. CHARALAMBIDES, S. CLAYMAN et G. PAVLOU. 2015, «Adaptive resource management and control in software defined networks», *IEEE Transactions on Network and Service Management*, vol. 12, nº 1, doi:10.1109/TNSM.2015.2402752, p. 18–33, ISSN 1932-4537.

VALIANT, L. 2013, *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*, Basic Books, Inc., New York, NY, USA, ISBN 0465032710, 9780465032716.

VALIANT, L. G. 1984, «A theory of the learnable», *Commun. ACM*, vol. 27, nº 11, doi:10.1145/1968.1972, p. 1134–1142, ISSN 0001-0782. URL http://doi.acm.org/10.1145/1968.1972.

VAN MIEGHEM, P. et F. A. KUIPERS. 2004, «Concepts of exact qos routing algorithms», *IEEE/ACM Transactions on Networking*, vol. 12, nº 5, doi:10.1109/TNET.2004.836112, p. 851–864, ISSN 1063-6692.

WANG, G., Y. ZHAO, J. HUANG, Q. DUAN et J. LI. 2016, «A k-means-based network partition algorithm for controller placement in software defined network», .

WANG, G., Y. ZHAO, J. HUANG et W. WANG. 2017, «The controller placement problem in software defined networking: A survey», *IEEE Network*, vol. 31, nº 5, doi:10.1109/MNET.2017.1600182, p. 21–27, ISSN 0890-8044.

WANG, G., Y. ZHAO, J. HUANG et Y. WU. 2018, «An effective approach to controller placement in software defined wide area networks», *IEEE Transactions on Network and Service Management*, vol. 15, nº 1, doi:10.1109/TNSM.2017.2785660, p. 344–355, ISSN 1932-4537.

WWW.DEVCONF.RU. 2015, «Watcher, the infrastructure optimization service for openstack», cahier de recherche.

XIAO, P., W. QU, H. QI, Z. LI et Y. XU. 2014, «The sdn controller placement problem for wan», dans *2014 IEEE/CIC International Conference on Communications in China (ICCC)*, ISSN 2377-8644, p. 220–224, doi:10.1109/ICCChina.2014.7008275.

XIE, H. et M. ZHANG. 2013, «Parent selection pressure auto-tuning for tournament selection in genetic programming», *IEEE Transactions on Evolutionary Computation*, vol. 17, nº 1, doi:10.1109/TEVC.2011.2182652, p. 1–19, ISSN 1089-778X.

YAO, G., J. BI, Y. LI et L. GUO. 2014, «On the capacitated controller placement problem in software defined networks», *IEEE Communications Letters*, vol. 18, nº 8, doi:10.1109/LCOMM.2014.2332341, p. 1339–1342, ISSN 1089-7798.

YAO, L., P. HONG, W. ZHANG, J. LI et D. NI. 2015, «Controller placement and flow based dynamic management problem towards sdn», dans *2015 IEEE International Conference on Communication Workshop (ICCW)*, ISSN 2164-7038, p. 363–368, doi:10.1109/ICCW.2015.7247206.

YAP, K.-K., M. MOTIWALA, J. RAHE, S. PADGETT, M. HOLLIMAN, G. BALDUS, M. HINES, T. KIM, A. NARAYANAN, A. JAIN, V. LIN, C. RICE, B. ROGAN, A. SINGH, B. TANAKA, M. VERMA, P. SOOD, M. TARIQ, M. TIERNEY, D. TRUMIC, V. VALANCIUS, C. YING, M. KALLAHALLA, B. KOLEY et A. VAHDAT. 2017, «Taking the edge off with espresso: Scale, reliability and programmability for global internet peering», dans *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM 17.

YEGANEH, S. H., A. TOOTOONCHIAN et Y. GANJALI. 2013, «On scalability of software-defined networking.», *IEEE Communications Magazine*, vol. 51, nº 2, p. 136–141. URL http://dblp.uni-trier.de/db/journals/cm/cm51.html#YeganehTG13.

YUAN, T., X. HUANG, M. MA et J. YUAN. 2018, «Balance-based sdn controller placement and assignment with minimum weight matching», dans *2018 IEEE International Conference on Communications (ICC)*, ISSN 1938-1883, p. 1–6, doi:10.1109/ICC.2018.8422637.

ZHANG, Q. et H. LI. 2007, «Moea/d: A multiobjective evolutionary algorithm based on decomposition», *IEEE Transactions on Evolutionary Computation*, vol. 11, nº 6, doi:10.1109/TEVC.2007.892759, p. 712–731, ISSN 1089-778X.

ZHANG, T., A. BIANCO et P. GIACCONE. 2016, «The role of inter-controller traffic in sdn controllers placement», dans *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, p. 87–92, doi:10.1109/NFV-SDN.2016.7919481.

ZHANG, Y., N. BEHESHTI et M. TATIPAMULA. 2011, «On resilience of split-architecture networks», dans *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, ISSN 1930-529X, p. 1–6, doi:10.1109/GLOCOM.2011.6134496.

ZHOU, A., B.-Y. QU, H. LI, S.-Z. ZHAO, P. SUGANTHAN et Q. ZHANG. 2011, «Multiobjective evolutionary algorithms: A survey of the state of the art», *Swarm and Evolutionary Computation*, vol. 1, doi:10.1016/j.swevo.2011.03.001, p. 32–49.

ZHU, Y. et M. H. AMMAR. 2006, «Algorithms for assigning substrate network resources to virtual network components.», dans *INFOCOM*, IEEE, ISBN 1-4244-0221-2. URL `http://dblp.uni-trier.de/db/conf/infocom/infocom2006.html#ZhuA06`.

ZITZLER, E., M. LAUMANNS et L. THIELE. 2001, «SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization», dans *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, édité par K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou et T. Fogarty, International Center for Numerical Methods in Engineering, Athens, Greece, p. 95–100.

Chapitre 3

Propositions pour une architecture SDN flexible et distribuée

“ He took his vorpal sword in hand: Long time the manxome foe he sought – So rested he by the Tumtum tree, And stood awhile in thought”

Lewis Carroll

Sommaire

3.1 Introduction

Au cours des dernières années, la communauté de recherche autour des réseaux a connu une période d'intense activité qui a donné lieu à l'émergence de différentes architectures ou paradigmes comme le SDN. La centralisation (logique ou physique) du plan de contrôle, devait permettre d'apporter aux applications réseaux la flexibilité attendue et permettre de répondre à de nombreux cas d'utilisations concrets.

Pour maîtriser la complexité dans ces architectures, l'idée était le développement de vues proposant une abstraction du réseau qui permettent le développement d'applications agissant au travers de contrôleurs SDN. L'ajout de nouveaux services ou le remplacement de services existants ou encore plus généralement la gestion de la vie du service doit aussi être une caractéristique inhérente aux architectures s'appuyant sur le concept de SDN.

De fait, l'introduction d'un service nouveau doit à priori pouvoir être réalisé simplement en ajoutant une nouvelle application qui interagirait directement avec le contrôleur réseau. Dans ces conditions il n'est plus nécessaire de modifier des éléments en charge du plan de données dont on attend avant tout de bonnes performances de transfert.

Les architectures centralisées présentent cependant des inconvénients. L'un d'eux réside dans le déploiement statique d'un contrôleur ou de contrôleurs dans le réseau, alors que leur placement et leur nombre sont déterminants pour les performances attendues.

Comme les performances sont essentiellement des caractéristiques inhérentes aux applications mises en œuvre, cela limite la capacité d'adaptation de ces architectures pour réaliser des services à la demande avec des SLAs variés.

D'autre part, la conception des contrôleurs s'apparente généralement à un dispositif de type driver (par analogie avec un OS informatique) qui est connecté au réseau. On parle d'ailleurs de Network OS. Dans ces conditions, c'est l'application (c'est-à-dire l'élément intelligent de l'architecture qui contrôle le service) qui agit effectivement sur le réseau au travers du filtre que constitue le contrôleur-driver.

Mais cela nécessite que le contrôleur dispose d'une connaissance fine et permanente de ce qui se passe dans le réseau avec des fonctions de monitoring élaborées car sinon il y a le risque d'introduire des incohérences, en particulier dans les environnements virtualisés partagés entre plusieurs opérateurs.

Enfin, les différentes applications peuvent agir au travers des contrôleurs sur les mêmes ressources et sur les mêmes dispositifs, et introduire dans le réseau des ensembles de règles qui ne sont pas nécessairement cohérentes entre elles ce qui peut aussi induire des effets non prévus.

Il n'est pas surprenant dans ces conditions que les réseaux aient adopté dès le départ dans leur conception une approche distribuée car c'était l'approche la plus flexible, la plus simple et la plus pertinente; l'objectif principal étant de disposer de réseaux résilients et fiables.

En résumé, le risque est que l'architecture centralisée bien qu'offrant des avantages potentiels réduise à néant les différentes plus values des architectures réseaux distribuées.

Dans la partie état de l'art, nous avons présenté différentes propositions et contributions dont l'objectif vise à surmonter un certain nombre de limitations propres aux architectures SDN qui sont essentiellement basées sur OF.

Ces inconvénients sont liés en premier lieu à la centralisation des fonctions de contrôle, en second lieu à l'absence de programmabilité des SOF, (la programmabilité étant déportée pratiquement totalement sur le contrôleur central), en troisième lieu à une conception statique du déploiement des dits services et finalement à un manque de flexibilité

du protocole OF lui même qui est organisé autour de la notion de table de flots.

Si l'on prend un peu de hauteur, le cœur de la problématique est sous-jacent au passage d'une architecture distribuée qui prévalait jusqu'à présent dans les réseaux et a fait ses preuves, à une architecture centralisée dite SDN dans laquelle les traitements intelligents sont exécutés par un organe central.

Parmi les inconvénients de l'approche distribuée qui définissent les promesses de l'architecture centralisée, on trouve par exemple le fait que des services réseaux élaborés de type (pare-feu ou DPI "Deep Packet Inspection" ou autres) doivent être réalisés à partir de dispositifs intermédiaires (*middle-boxes*) statiques qui sont disposés dans le réseau et vers lesquels transitent les flux. On trouve aussi, le temps de réaction nécessaire pour réaliser certaines actions, par exemple pour faire converger un service de routage après un changement de topologie. Un troisième inconvénient flagrant est la stratification du réseau, qui est très difficile à faire évoluer.

Nous avons essayé de résumer ces réflexions dans le tableau référencé 3.1 où nous comparons les avantages et inconvénients d'une approche centralisée par rapport à une approche distribuée.

Comme cela est illustré dans la partie état de l'art, de nombreuses propositions ont été faites pour réduire où lever les limitations des architectures SDN centralisées basées sur OF. On peut citer comme thématiques parmi les propositions déclinées de multiples manières dans les publications:

- La distribution physique des contrôleurs dans le réseau via une interface est-ouest.
- L'introduction de logiques de traitement locaux au niveau des SOFs, permettant de réduire l'impact de la centralisation.
- La notion de contrôleur physiquement distribué et logiquement centralisé, plutôt dans un contexte de datacenters.
- L'introduction de contrôleurs distribués mais avec une organisation hiérarchique.
- La séparation des applications réseaux en deux niveaux ou davantage traités par des contrôleurs organisés hiérarchiquement, en fonction de la localité du traitement.
- Des mécanismes de redéploiement de l'attachement des SOFs aux contrôleurs en fonction du trafic,
- L'association de contrôleurs en clusters avec des mécanismes de redondance, permettant d'augmenter la fiabilité.
- Des mécanismes de déploiement et redéploiement dynamiques de contrôleurs.

Ces différentes propositions ne traitent en général qu'une partie des problématiques posées et ne permettent pas de résoudre le problème dans son ensemble. Les limitations de l'approche OF sont trop contraignantes et il nous a semblé, pour que le paradigme SDN reste valide, qu'il fallait envisager une approche différente. En réalité il est probable que OF ne convient surtout qu'au contexte du datacenter qui a pour caractéristiques des bandes passantes très élevées et une certaine localité des fonctions avec par exemple son intégration dans le module OpenStack (URMILA et POL [2014]). Mais ce protocole n'est pas adapté au contexte des réseaux des opérateurs Telco qui sont distribués sur de grandes aires géographiques avec des nœuds connectés par des liens avec des débits hétérogènes.

	Centralisation	Distribution
Pros	Une vue globale permet une optimisation globale Reconfiguration rapide et facile Mise jour facile des applications Forte potentialité pour implémenter des services réseaux	Des mécanismes sont disponibles pour les traitements locaux Résilience du réseau Des algorithmes distribués peuvent permettre une optimisation
Cons	Single Point Of Failure (SPOF) Délais pour descendre de nouvelles règles dans le réseau La centralisation introduit des goulots d'étranglements de trafic Besoin d'un protocole de contrôle SDN centralisé Le protocole OF est peu flexible et difficile à faire évoluer. Le déploiement statique des contrôleurs limite le SLA des services Cohabitation de multiples applications sans assurance de cohérence entre elles	Délais de convergence en cas de panne Pas de vision globale rend plus difficile une optimisation globale Besoin d'un protocole de contrôle distribué Des dispositifs intermédiaires (*middle boxes*) sont nécessaires pour implémenter des services spécifiques

Table 3.1 – Avantages et inconvénients des architectures centralisées et distribuées

C'est un des motifs de la proposition SDN 2.0 initiée par Scott Shenker, un des initiateurs de l'approche SDN [1]. Dans cette proposition, quelques principes sont proposés: (1) une distinction claire entre périphérie du réseau SDNisable par rapport au réseau cœur qui reste basé sur les technologies classiques, (2) la notion de middle-boxes virtuelles ou physiques, (3) l'ouverture des services réseaux à des utilisateurs tiers.

Ce constat a aussi amené la conversion relativement récente chez les industriels du concept de SDN en SD-WAN (Software Defined WAN).

L'idée générale du SD-WAN est de réduire les ambitions du SDN appliqué au WAN à des approches visant à améliorer l'automatisation des services et surtout à réaliser une correspondance fine entre les applications qui s'exécutent dans les datacenters et les mécanismes mis en œuvre au niveau du WAN pour assurer QoS, sécurité, et réduire les

[1] `https://www.sdxcentral.com/articles/news/scott-shenker-preaches-revised-sdn-sdnv2/2014/10/`

goulots d'étranglements en optimisant le trafic [ALI et collab., 2017].

Les réseaux B2 et B4 de Cisco correspondent déjà à une approche de ce type plus réaliste et plus crédible.

3.2 DICES, Une architecture SDN orientée services

En prenant en compte les observations qui précèdent, notre suggestion a été de proposer une architecture SDN appelée DICES [2] dans laquelle les applications réseaux sont vues comme des services auxquels est associé un certain SLA. Il y a une multiplicité de services réseaux. On peut citer comme exemple un service basique de routage, une application de pare-feu, une application de re-routage rapide, ou enfin une application de surveillance ("monitoring").

Dans les architectures patrimoniales, ces services sont généralement localisés (pare-feux par exemple) dans des "middlebox", et parfois partiellement ou complètement distribués comme des sondes pour la surveillance quand cela est possible, ou pour le routage.

Une architecture orientée services réseaux, doit pouvoir permettre de déployer ces services en prenant en compte des critères de performances rassemblés sous le terme de SLA. La distribution du service dans la topologie, sa localité, le placement des fonctions qui le réalisent dans la topologie vont permettre d'atteindre tel ou tel niveau de SLA en prenant aussi en compte les contraintes propres au service.

Il faut donc une architecture qui permette de déployer des services de manière automatique avec différents niveaux de localité correspondant aux différents SLA.

Nous voulons ensuite que notre architecture permette de définir des services réseaux de manière aussi flexible que possible et qu'ils puissent être déployés et mis à jour rapidement et facilement.

Nous souhaitons en résumé que l'on puisse gérer de manière automatisée le cycle de vie du service, c'est-à-dire modéliser et déployer le service réseau, puis faire des mises à jour éventuellement à chaud et sans coupure, ou après interruption et redémarrage du service.

Pour réaliser cela, nous avons défini quatre briques essentielles:

- un orchestrateur réseau qui est un élément fondamental responsable de la validation et du déploiement du service et plus généralement de la gestion du cycle de vie des services,

- un processus de composition de modèles de services pour construire des services réseaux complexes à partir de composants élémentaires et d'un langage de description des services réseau,

- Des éléments réseaux génériques dans lesquels peuvent être déployés et exécutés les modèles de services réseaux élaborés par l'orchestrateur.

- Une fonction contrôleur supportée par les nœuds réseau. Cette fonction peut être centralisée, distribuée ou à plat avec une gradation possible du contrôleur élémentaire qui ne gère que le plan de contrôle local du nœud lui même à un contrôleur centralisé qui gère le plan de contrôle d'un ensemble de nœuds.

[2] Dynamic adaptIve serviCe-drivEn SDN architecture

Ces briques permettent de réaliser différents points clés de notre proposition.

Le remplacement des SOF par le concept de nœuds réseaux banalisés génériques programmables qui leur permet de supporter aussi bien la programmation de fonctions de contrôle que de fonctions de traitement des paquets de données.

L'utilisation de micro-services réseaux banalisés élémentaires que l'on peut composer, via un langage de composition de manière à réaliser un modèle du service. Ce modèle peut-être déployé et exécuté au niveau des nœuds réseau aussi bien pour la partie contrôle que la partie traitement des données. L'orchestrateur dispose d'un outil de vérification du modèle avant déploiement.

Un protocole de déploiement et de mise à jour du service, permet de déployer les services via un canal de communication comparable au protocole OF. Il est aussi doté de fonctionnalités plus riches intégrant à la fois gestion et contrôle, depuis l'orchestrateur vers les éléments réseau.

Dans la suite, chaque brique de la proposition est détaillée. Puis nous nous concentrons sur la modélisation et le déploiement des services en décrivant le processus dynamique de fourniture de services.

Le concept est ensuite illustré via un cas d'usage de re-routage rapide.

3.2.1 Description de l'architecture proposée

L'architecture proposée est un concept de réseau SDN, et est illustrée sur la figure 3.1. Cette architecture réseau est constituée des entités et fonctions suivantes:

1. Un orchestrateur dont le but est d'assembler, de déployer et d'exécuter, en d'autres termes de gérer le cycle de vie de services réseaux qui sont déployés.

2. Des services réseaux qui sont modélisés et composés à partir de µ-modèles élémentaires décrivant des services réseaux aussi élémentaires que possible. Les modèles de services de base sont stockés dans une base de données.

3. Un langage de modélisation des services qui permet de composer les µ-services réseaux élémentaires.

4. Des éléments réseau simplifiés, génériques et performants en charge du plan de transmission et du plan de contrôle. Ces éléments réseau sont programmables aussi bien du point de vue du plan de contrôle que du plan de données. Les modèles de services (plan de contrôle ou plan de données) sont déployés pour être exécutés sur ces éléments réseaux.

Dans le modèle, la distribution et la localisation des contrôleurs SDN dans le réseau est déterminée dynamiquement par les contraintes et les caractéristiques des services réseau qu'ils doivent exécuter.

Les contrôleurs peuvent donc être déployés sur un ou des éléments réseaux. Ces éléments réseaux pourront jouer le rôle d'un contrôleur centralisé pour l'exécution du service sur l'ensemble du réseau ou d'un contrôleur plus local dans une architecture de contrôleurs distribués.

Ce principe permet des contrôleurs organisés hiérarchiquement de manière à séparer l'exécution de la partie locale de la partie globale du plan de contrôle.

Les modèles qui décrivent les services élémentaires réseaux sont stockés dans une bibliothèque qui peut être sollicitée par l'orchestrateur de services. En fonction des demandes des clients, l'orchestrateur de services assemble les différents modèles élémentaires en modèles de services globaux. Le processus de composition prend en compte le

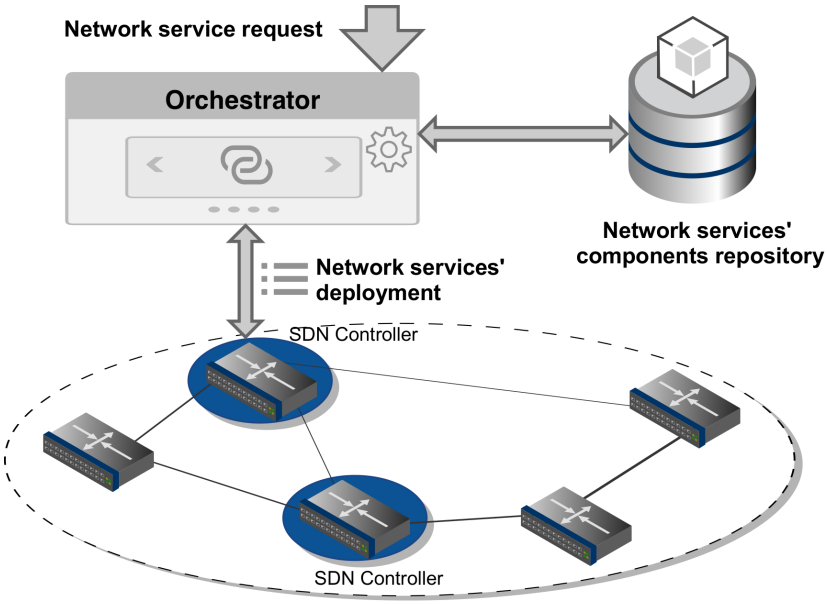


Figure 3.1 – Vue globale de l'architecture réseau

SLA demandé et les contraintes des topologies réseaux afin de construire des modèles de services réseau de bout en bout.

Une fois le modèle de service réseau assemblé, la cohérence du service réseau est vérifiée et validée par l'orchestrateur.

Après l'étape de validation du modèle du service, les contrôleurs sont répartis et déployés dans le réseau de manière à exécuter de manière optimale les services pris en charge (c'est-à-dire comme prévu par le SLA).

Pour cela, des algorithmes sont conçus pour déployer le nombre optimal de contrôleurs avec leur emplacement. Ensuite, les contrôleurs SDN sont activés dans le réseau et associés aux éléments du réseau qu'ils doivent contrôler.

La dernière étape consiste à déployer les services réseau par l'orchestrateur dans les contrôleurs réseau.

Les services réseau sont exécutés localement par les contrôleurs qui pilotent les éléments du réseau.

Pour réaliser cette fonction, chaque contrôleur expose un ensemble d'APIs qui permettent l'exécution des modèles de service via un moteur d'exécution.

La logique que nous proposons est illustrée dans la figure 3.2.

La proposition est organisée autour de trois fonctionnalités principales suivantes :

- Tout d'abord, la nécessité de pouvoir définir de manière flexible et dynamique des services réseau s'appuyant sur des modèles, qui décrivent les services réseau élémentaires, utilisés et assemblés par l'orchestrateur pour composer le service ciblé. Une fois assemblé, le service réseau est testé et validé.

- Deuxièmement, une fonction de contrôle SDN générique disponible et prête à être instanciée et activée dans tous les éléments du réseau. Chaque noeud du réseau intègre cette fonction de contrôleur, qu'elle soit utilisée ou non. Lorsqu'il est activé, le contrôleur permet d'exécuter les services réseau dans le réseau en fonction de son domaine d'action. Ainsi, le contrôleur peut être à l'état de repos puisqu'il ne contrôle aucun élément de réseau, mais il peut également contrôler un seul élément de réseau (l'élément de réseau qui intègre le contrôleur), ou il peut contrôler un

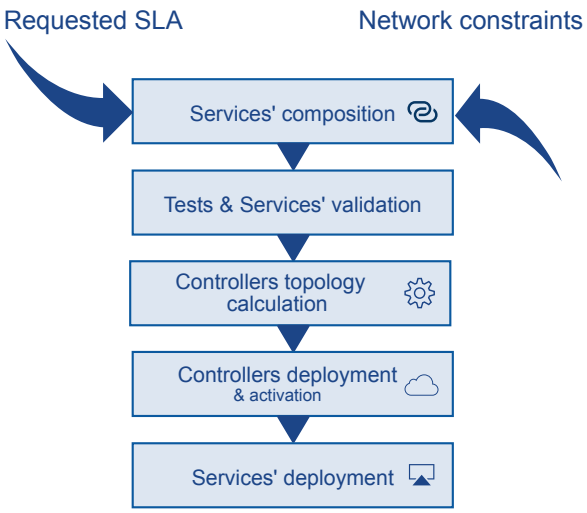


Figure 3.2 – Etapes de déploiement d'un service réseau

ensemble d'éléments de réseaux pour exécuter un service applicable à un ensemble d'éléments de réseau. comme cela peut être le cas pour un service de fast-rerouting.

- Troisièmement, une topologie adaptative des contrôleurs SDN qui sont organisés de manière hiérarchique et qui sont activés dans le réseau par rapport au SLA attendu pour les services réseau.

  La possibilité d'optimiser et d'adapter la topologie des contrôleurs dans le réseau permet de réaliser les différents SLA de services réseau. Elle permet aussi de répondre à la contrainte d'évolutivité tout en assurant la flexibilité. A cet effet, des algorithmes basés sur la mesure de la centralité des nœuds à l'aide de différentes métriques pourront calculer la topologie des contrôleurs du réseau par rapport aux services réseau et aux performances attendues utilisées comme paramètres d'entrée pour ces algorithmes.

3.2.2 Une organisation adaptative et distribuée de contrôleurs

Dans le modèle réseau que nous proposons, les contrôleurs sont répartis de manière optimale dans le réseau afin d'exécuter les modèles de services réseau poussés par l'orchestrateur dans les contrôleurs et dans la partie plan de données des éléments réseaux.

L'emplacement des contrôleurs dans le réseau ainsi que le nombre de nœuds du réseau à contrôler sont optimisés en fonction de différents paramètres. Les paramètres suivants sont à prendre en compte dans l'optimisation :

- Les catégories de services réseaux ciblées et leurs performances attendues,
- Le SLA requis pour le service,
- Les paramètres liés à la topologie du réseau, par exemple, le nombre de nœuds à contrôler, les ressources des nœuds, les latences des liens entre les nœuds et leur bande passante, les performances attendues en terme d'écoulement du trafic, la connectivité entre les nœuds, ...

- Les différents domaines techniques ou administratifs du réseau à gérer.

L'emplacement optimal des contrôleurs réseaux dépend donc des services réseaux implémentés. Afin d'atteindre les performances visées, il est nécessaire pour certains services d'être exécutés localement à proximité des ressources matérielles, ce qui peut nécessiter un nombre important de contrôleurs.

D'autres services réseaux peuvent avoir besoin d'une vue d'ensemble du réseau afin d'être optimisés. Par exemple, la fonction de fast re-routing nécessite d'être traitée localement au moins en partie afin de diminuer le délai d'exécution, et d'identifier rapidement la liaison ou le nœud défectueux et d'activer une liaison de secours.

Cependant, si l'objectif est de ré-acheminer le trafic de manière optimale, en fonction de certains critères (comme par exemple la latence), il faut avoir accès à une vision plus globale du réseau et de sa topologie.

Pour cela, l'application doit être exécutée au-dessus d'un contrôleur de niveau supérieur qui contrôle un plus grand nombre de nœuds et qui dispose d'une vision plus globale du réseau.

D'autres services réseau, tels que par exemple un service d'ingénierie de trafic, dont l'objectif est d'écouler le trafic de manière optimale, nécessiteront moins de réactivité mais auront besoin impérativement d'une vision globale du réseau. L'application devra disposer d'informations sur tous les nœuds et liens et, grâce à cette vue globale, pourra calculer les chemins optimaux pour écouler le trafic. Elle pourra s'exécuter sur le contrôleur de niveau supérieur.

3.2.3 Organisation des éléments réseau

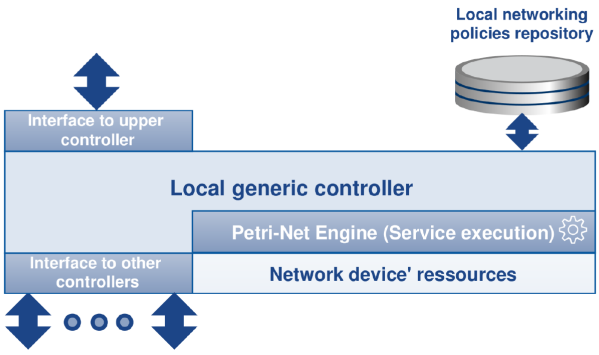


Figure 3.3 – Architecture fonctionnelle d'un élément réseau

Dans notre proposition, une fonction de contrôle générique (c'est-à-dire un agent contrôleur) est intégrée dans chaque nœud du réseau. En ce qui concerne les nœuds de réseau legacy, on peut imaginer la mise en œuvre d'un agent de contrôle externe permettant la compatibilité avec les équipements de réseau existants.

En principe, le contrôleur contrôle les ressources réseau (par exemple, les tables de routage) qui lui sont directement rattachées sur le nœud qui le supporte. Mais par ailleurs, il peut contrôler un groupe donné d'éléments réseaux en fonction des services réseau qu'il doit implémenter.

L'implémentation des services réseau s'effectue une fois que la topologie des contrôleurs SDN est composée et déployée sur les nœuds réseau associés.

Le contrôleur est également capable d'échanger avec un contrôleur de niveau supérieur, afin, par exemple, de renvoyer des informations de topologies relatives aux nœuds dont il a la responsabilité ou concernant la supervision des services ou des ressources, si nécessaire.

Les services réseau sont poussés et activés depuis le contrôleur de niveau supérieur ou depuis l'orchestrateur. Ils sont exécutés par les contrôleurs de niveau inférieur à l'aide d'un moteur d'exécution de service tel que illustré dans la figure 3.3.

3.2.4 Une organisation hiérarchique et distribuée des contrôleurs

La capacité de mise à l'échelle (scalability) de ces architectures avec des contrôleurs centralisés est un autre point clé. Cette propriété entraîne un besoin de distribution des contrôleurs dans le réseau avec une disposition hiérarchique à au moins deux niveaux. Ceci est illustré par la figure 3.4.

En fonction des applications à déployer et de la taille du réseau, il peut y avoir des configurations dans lesquelles un seul nœud de contrôle peut concentrer une partie importante du trafic, ce qui a un impact sur la capacité de mise à l'échelle de l'architecture et, par conséquent, du service.

Une disposition hiérarchique avec un contrôleur central et des nœuds contrôlant un sous-ensemble du réseau offrent de meilleures propriétés de mise à l'échelle. De plus, le contrôleur central facilitera l'exécution des services qui nécessitent une vue globale du réseau tandis que les contrôleurs locaux permettront l'exécution des services qui doivent être proches des nœuds du réseau en raison de contraintes strictes de SLA.

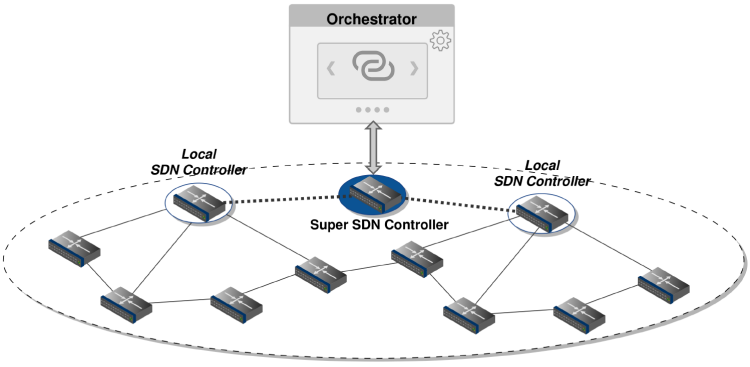


Figure 3.4 – Architecture hiérarchique de contrôleurs

3.2.5 Séparation des domaines

Les différents domaines techniques et administratifs du réseau pourront reposer sur une organisation hiérarchique des contrôleurs avec au moins deux niveaux : des contrôleurs feuilles pour un domaine et un contrôleur de niveau supérieur qui a une vue globale inter-domaines.

A l'intérieur d'un réseau de télécommunication, on peut distinguer notamment le domaine réseau cœur, le domaine réseau d'agrégation, le domaine réseau d'accès, le domaine Datacenter, le domaine réseau d'entreprise. On peut également distinguer les différents systèmes autonomes sur lesquels une vision globale est recherchée afin d'optimiser le trafic inter-AS.

Le contrôleur inter-domaines gère une abstraction de niveau supérieur qui permet d'effectuer des optimisations transversales inter-domaines.

3.2.6 Provisionnement dynamique des services

Modélisation des services avec les réseaux de Petri

A titre d'illustration, on peut citer quelques exemples de services réseau. Il peut s'agir, par exemple, d'un service de connectivité de bout en bout, d'un service VPN de niveau 2 ou 3, d'un service de routage de trafic, de la supervision ou de la surveillance de segments et d'équipements de réseau.

Ils sont construits à l'aide de services réseau élémentaires qui peuvent être composés en un macro-service réseau.

Les approches SDN visent, en particulier, à rendre des services réseau flexibles et programmables. Pour ce faire, nous nous appuyons sur un modèle de service réseau élémentaire suffisamment abstrait, c'est-à-dire indépendant des types de ressources et des technologies sous-jacentes.

Les différents éléments peuvent ensuite être assemblés afin de construire des services réseau de bout en bout. Le résultat de cette composition est un service réseau qui peut être implémenté, déployé et exécuté sur le réseau. La méthode de modélisation des services réseau doit posséder les propriétés suivantes :

- Elle doit offrir un niveau d'abstraction suffisant c'est-à-dire être indépendante des technologies sous jacentes.
- Elle doit avoir des propriétés de modularité et de localité, ce qui signifie qu'elle doit offrir la possibilité de composer facilement ensemble des modules élémentaires effectuant des traitements pour créer des composants plus complexes tout en masquant leur complexité résultante.
- Elle doit offrir de bonnes propriétés d'encapsulation. Cela nécessite que les modules élémentaires des services réseau soient indépendants (couplage lâche) et génériques (indépendants des infrastructures). De plus, ces modules doivent être dotés d'interfaces standard et de propriétés d'encapsulation permettant d'obtenir un modèle de type boîte noire.
- Il faut pouvoir proposer des outils et des propriétés qui permettent de tester et de valider le modèle de service afin d'accélérer l'optimisation et les évolutions possibles.
- Il devrait être possible de convertir le modèle en une forme exécutable dans les contrôleurs réseau via un moteur d'exécution dédié.

Dans ce contexte, nous avons identifié les réseaux de Petri comme pertinents en tant qu'approche de modélisation par comparaison avec d'autres outils comme les machines à états et les chaînes de Markov. Un des intérêts des réseaux de Petri est en particulier qu'ils permettent de modéliser des composants élémentaires et de les assembler en composants plus large en permettant différents niveaux d'imbrication. Ils constituent également une alternative intéressante et puissante aux machines à états finis en termes d'expressivité des processus modélisés [MURATA, 1989].

Les réseaux de Petri permettent de modéliser des services complexes avec des contraintes de temps réel, avec de la concurrence et des interactions synchronisées entre les

processus. Ceci est particulièrement intéressant pour les services réseau dans lesquels le partage des ressources et le parallélisme entre processus sont des éléments clés.

D'autre part, les réseaux de Petri permettent de modéliser des services déterministes et non déterministes distribués alors que les machines d'états sont souvent plus limitées pour faire face à ce type de situations.

Il existe un certain nombre d'extensions (réseaux de Pétri stochastiques, colorés ou à files d'attente, réseaux avec des arcs inhibiteurs, ...) ainsi que différentes variantes telles que des réseaux de Pétri synchrones ou temporels qui introduisent des durées d'événements. Ces différentes extensions enrichissent énormément les possibilités de modélisation. Parmi ces extensions en particulier la notion d'arc inhibiteur [ANNIE, 2006] donnent aux réseaux de Pétri et à leurs extensions le pouvoir d'expression d'une machine de Turing [A JAVAN, 2013].

Finalement, les réseaux de Pétri hiérarchiques et imbriqués offrent une syntaxe qui permet d'obtenir la modularité nécessaire à la modélisation de services complexes [BRUNO et collab., 1994].

Propriétés On peut déduire d'un modèle à base de réseaux de Petri des propriétés intéressantes et utiles des systèmes étudiés et du modèle. Les propriétés sont généralement valides pour un marquage initial donné.

On peut aussi parfois déduire ces propriétés pour une famille de marquages en particulier à partir de techniques d'algèbre linéaire en s'appuyant sur certains invariants du réseau indépendants du marquage initial.

Les propriétés les plus connues sont:

- L'absence de blocage: il y a toujours des transitions franchissables,
- Quasi vivacité: Toute transition est franchissable au moins une fois,
- Vivacité: Toute transition est toujours accessible et franchissable dans la vie du système,
- Marquage d'accueil: on peut toujours revenir à l'état initial (initialisation du système)
- Réseau borné: Le nombre de marquages accessibles est fini.

Cycle de vie du service

Lors du déploiement, l'orchestrateur active les contrôleurs du réseau en fonction des contraintes des services et de la topologie du réseau. Ensuite, le modèle à base de réseaux de Petri des services est encodé dans un format de fichier de type XML et est déployé par l'orchestrateur dans les contrôleurs.

Au sein des contrôleurs, l'exécution des services est réalisée à l'aide d'un moteur d'exécution du modèle.

[MORENO et SALCEDO, 2008] proposent sur ce principe, par exemple Petri-net, qui permet une interaction avec des fonctions de bas niveau en utilisant un ensemble d'APIs adhoc.

La gestion du cycle de vie des services est abordée dans notre architecture. Le cycle de vie du service comprend la création, la modification ou la mise à jour du service, la surveillance et la mise hors service du service comme présenté par [AFLATOONIAN et collab., 2014].

La modélisation du service réseau sous forme d'un réseau de Petri et l'utilisation du modèle de réseau de Petri pour l'implémentation du service sur le contrôleur nous donnent l'opportunité de fournir une véritable implémentation dynamique de la gestion de la vie du service.

Par exemple, la mise à jour du service peut se faire facilement par une simple mise à jour du modèle de service, puis en exécutant les différentes étapes déjà définies de la validation jusqu'au déploiement.

Le principal problème qui pourrait survenir serait lorsque la mise à jour de la politique de service nécessite le déploiement d'un autre contrôleur (ou plusieurs) pour répondre à une nouvelle politique de service ou à de nouvelles contraintes réseau.

Dans ce cas, nous proposons de gérer la mise à jour du service en plusieurs étapes. Lors de la première étape, l'exécution du service est déléguée au contrôleur supérieur et l'instanciation de service peut être supprimée des contrôleurs inférieurs. Le SLA du service peut-être alors temporairement dégradé.

Lors de la deuxième étape, un nouvel ensemble de contrôleurs peut être déployé correspondant au nouveau SLA mais l'ancien service fonctionne toujours sur le contrôleur supérieur, éventuellement avec un SLA réduit temporaire. Le nouveau service avec son nouvel SLA est alors déployé sur le nouvel ensemble de contrôleurs et le service peut être commuté du contrôleur supérieur vers les contrôleurs inférieurs.

Concernant le monitoring du service, la fonction de monitoring du service doit être incluse dès le début de la définition du service et incluse dans l'étape de modélisation du service. A la fin, le service déployé intégrera ses fonctions de monitoring associées qui seront déclinées aux différents niveaux de contrôleurs.

3.2.7 Description d'un cas d'usage

Nous avons choisi comme exemple un service de re-routage rapide réactif implémenté dans un réseau Internet Protocol (IP) ou MPLS.

Le but de ce service est de mettre en place le plus rapidement possible un nouveau chemin de secours avec des caractéristiques acceptables pour dériver le trafic en cas de défaillance d'un nœud ou d'une liaison. Le chemin de secours n'est pas pré-établi.

La politique de service définira ici le délai maximal d'établissement d'un nouveau chemin pour dériver le trafic, la bande passante demandée et d'autres paramètres à respecter.

Le délai maximal de récupération est un problème critique dans le cas d'une implémentation SDN, où le contrôle peut être fortement centralisé.

La latence globale pour les échanges entre les SOFs et le contrôleur, les goulots d'étranglements liés aux variations de trafic peuvent retarder à la fois la détection du service, la décision de déploiement ainsi que le déploiement des nouvelles routes.

Si le contrôleur est très éloigné des éléments du réseau, ou s'il y a un goulot d'étranglement sur le trafic de contrôle ou encore si le contrôleur centralisé doit réaliser trop de traitements parce qu'il reçoit du fait de la panne trop de demandes de traitements, la détection et le traitement d'une panne peut prendre trop de temps et le retard qui en résulte pour établir la nouvelle liaison ne sera pas acceptable.

Pour résoudre ce problème, le service réseau doit plutôt être déployé à proximité des éléments du réseau ou bien l'élément réseau doit inclure des mécanismes spécifiques pour accélérer le traitement qui le rendent non générique. Si ce n'est pas le cas, le service doit être déployé sur un contrôleur implémenté assez près des nœuds affectés par la panne afin d'exécuter le service de re-routage. De cette manière on peut espérer que le

service soit conforme aux contraintes de délais.

3.2.8 Modélisation du cas d'usage par un réseau de Petri

Le diagramme de la figure 3.5 montre un exemple de modélisation de service réseau par un réseau de Petri. Le service de re-routage rapide est considéré dans un réseau MPLS ou IP comme décrit dans le paragraphe précédent. Les jetons sont utilisés ici pour modéliser des routes alternatives (c'est-à-dire des flots IP) en tant que ressources disponibles dans le réseau et pour définir l'état initial.

Le service pourrait être modélisé à l'aide des concepts de réseaux de Petri hiérarchiques [FEHLING, 1993] où les places peuvent être fusionnées en places de fusion et les transitions peuvent être remplacées par des transitions de "substitution" pour représenter un ensemble de transitions. On peut ainsi réaliser une description fine et à plusieurs niveaux emboités du service.

Le diagramme ne montre ici qu'une première étape de la modélisation, où chaque place et chaque transition peuvent être elles-mêmes décomposées en un ensemble de places et de transitions grâce à un modèle de réseau de Petri plus élémentaire pour rentrer dans le détail de la description du service.

Les cinq principaux états impliqués dans le premier niveau de modèle que nous proposons sont : l'état de repos, l'état d'erreur, le nouvel état de calcul du chemin, l'état de création et l'état d'activation.

- L'état de repos correspond à l'état de fonctionnement courant de l'application,
- L'état d'erreur correspond à l'état de détection d'erreur,
- Le nouvel état de calcul du chemin, génère des ressources alternatives,
- L'état de création crée les routes alternatives sélectionnées,
- L'état d'activation active l'itinéraire créé.

Cet exemple a été validé à l'aide du logiciel de modélisation Tina pour les réseaux de Petri [BERTHOMIEU et VERNADAT, 2006].

Un graphe de marquage de l'analyse d'accessibilité a été réalisé pour valider et prouver que le modèle est borné (c'est-à-dire que le nombre de jetons dans les places est borné tout au long de l'évolution du marquage), vivant (c'est-à-dire que toutes les actions sont accessibles depuis tout marquage courant, le réseau de Petri est donc exempt de blocages) et réversible (un retour à l'état initial est toujours possible).

Une analyse structurelle, qui fournit des propriétés indépendantes du marquage initial, a également été effectuée avec l'outil Tina, montrant que le modèle n'est pas invariant (c'est-à-dire que des propriétés comme la vivacité, par exemple, peuvent changer avec le marquage initial) et montrant aussi la propriété de consistance (c'est-à-dire qu'il existe un marquage initial tel que chaque transition peut se produire).

Ces vérifications effectuées avec Tina sur des modèles à base de réseaux de Petri sont une première approche de ce que pourrait être un outil de vérification de la consistance et de la cohérence des services réseaux modélisés par ce moyen.

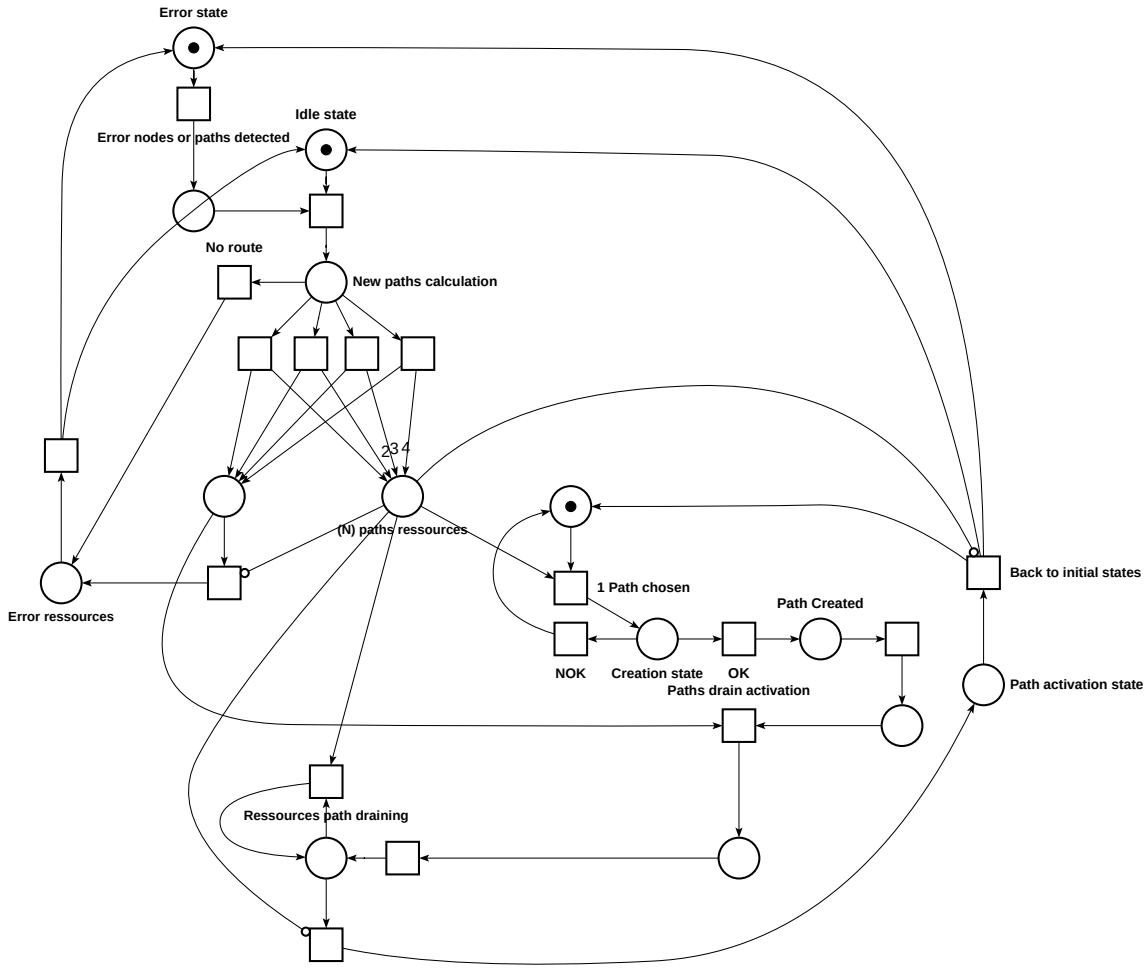


Figure 3.5 – Modélisation d'un Fast rerouting service par réseaux de Petri

3.2.9 Algorithme de déploiement des contrôleurs

Afin de consolider ce travail nous avons proposé un algorithme de déploiement de contrôleurs correspondant au cas d'usage étudié.

La question du déploiement optimal des contrôleurs est cependant étudiée de manière plus approfondie au chapitre suivant.

Comme expliqué ci-dessus, un algorithme est nécessaire pour le placement des contrôleurs dans le réseau afin de gérer et de s'adapter dynamiquement aux contraintes des services.

Dans le cas du service de fast-rerouting, l'un des principaux critères de localisation du contrôleur sera la latence entre les nœuds et leur contrôleur. Ce délai aura une incidence directe sur le délai de rétablissement du service, en effet, l'information indiquant une défaillance doit être envoyée au contrôleur pour déclencher un nouveau calcul d'itinéraire et pour le déploiement d'une route de secours. Ce délai doit être inférieur à un seuil donné pour garantir qu'une nouvelle liaison sera mise en place dans le respect du SLA du service. Ainsi, l'algorithme fourni doit au moins garantir ce critère principal lors de la mise en place des contrôleurs.

Pour ce cas d'utilisation particulier, nous avons imaginé un algorithme récursif qui exploite les métriques de plus court chemin entre les nœuds du graphe afin de choisir le nombre minimum de nœuds dans lesquels les contrôleurs seront activés.

En entrée, l'algorithme dispose du délai maximum toléré entre un nœud et son con-

trôleur, et fournit en sortie la distribution des contrôleurs associée à la liste des nœuds contrôlés pour respecter cette contrainte.

Les figures 3.6 et 3.7 illustrent une comparaison des performances calculées sur six topologies de réseaux réels extraites de la base zoo-topology [KNIGHT et collab., 2011].

Le critère considéré ici est le délai maximal de 50 ms normalisé de récupération d'une nouvelle route dans un réseau MPLS et qui est nécessaire pour établir une nouvelle liaison en cas de défaillance d'une liaison ou d'un nœud.

Deux cas sont simulés : le premier, basé sur une approche standard c'est-à-dire centralisée, qui ne considère qu'un seul contrôleur pour tous les nœuds. Le nœud supportant le contrôleur est sélectionné à l'aide du critère de centralité de proximité. Dans cette situation, il n'y a aucune garantie que le délai maximal de recouvrement sera atteint. (cf. figure 3.6)

Le deuxième cas utilise l'algorithme que nous proposons pour déterminer une distribution des contrôleurs permettant de respecter une contrainte de latence assurant une récupération en 50 ms. Le délai maximal entre un nœud et un contrôleur, qui est fourni en entrée de l'algorithme, est de 12.5 ms (figure 3.7).

Ce délai correspond à environ 50 ms pour la latence de récupération si l'on considère que : la durée de traitement globale est d'environ 10 ms, et le délai de détection de la panne est de 15 ms. Les diagrammes fournissent pour chaque nœud contrôleur : les latences minimale, médiane et maximale pour le délai de mise en œuvre d'une nouvelle route, et l'écart-type autour de la valeur moyenne.

La figure 3.7 montre que, dans tous les cas considérés, le délai maximum de récupération respecte la contrainte de service cible de 50 ms. Cette comparaison illustre le gain obtenu en utilisant la stratégie proposée pour le placement des contrôleurs.

Elle nécessite bien sûr plusieurs contrôleurs distribués et placés de manière optimisée pour réaliser le service avec son SLA et avec un nombre minimum de contrôleurs.

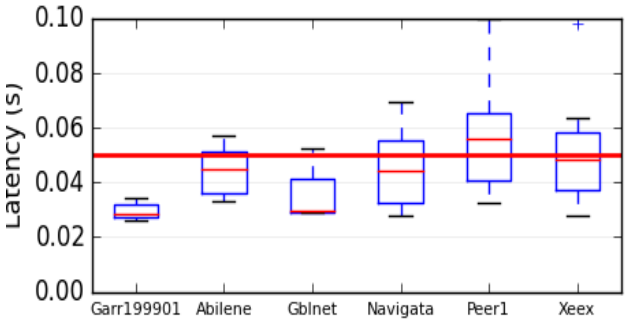


Figure 3.6 – Approche classique

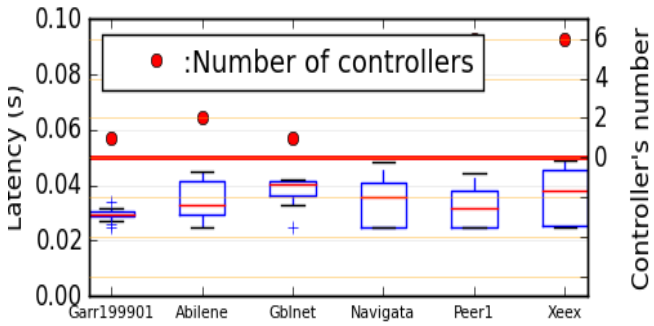


Figure 3.7 – Approche proposée

3.2.10 Conclusion

Dans ce travail, nous proposons la définition d' une architecture de réseau programmable de manière dynamique, flexible, évolutive et orientée services.

L'objectif de cette architecture est d'offrir aux réseaux de télécommunications plus de programmabilité et plus de flexibilité en utilisant le plan de contrôle centralisé et le paradigme SDN de softwarisation des réseaux. Nous proposons pour cela:

- La définition d'éléments de réseau génériques en charge du plan de données et du plan de contrôle. L' avantage de ces éléments est qu'ils sont génériques et

programmables pour supporter un plan de données et un plan de contrôle avec différents niveaux de localité. Ils intègrent, une fonction de contrôle générique disponible à l'intérieur de chaque élément réseau, qui peut être activée pour piloter des éléments de réseau sélectionnés afin d'exécuter les services réseau. Cela permet de déployer, de manière flexible, des contrôleurs en fonction des contraintes du réseau et de la performance des services attendus.

- Un processus de déploiement de contrôleurs piloté par les SLA des services, la topologie du réseau et les contraintes des dits services. La topologie des contrôleurs dans le réseau doit par conséquent être en ligne avec la performance des services et les contraintes du réseau.

- Un processus de modélisation des services réseau basé sur les réseaux de Petri, qui vise à couvrir un large éventail de services tout en offrant des mécanismes efficaces pour la composition et la validation des services. La structure et la composition des services est donc capable d'évoluer et de s'adapter en fonction des besoins des utilisateurs. Les réseaux de Petri sont aussi une première approche de ce que pourrait être un outil de vérification de la consistance et de la cohérence des services réseaux modélisés.

  Un moteur Petri-net inclus dans chaque contrôleur réseau permet d'exécuter les services réseau déployés. C'est l'élément de base nécessaire à l'exécution des services, au niveau du contrôleur, selon leur modèle.

Chacun de ces éléments est un élément clé d'un scénario de définition d'une architecture globale, et chacun d'entre eux nécessite un examen plus approfondi afin d'évaluer qualitativement les gains en termes de déploiement et de vitesse d'exécution des services, de performances en termes d'évolutivité, de réactivité et de QoS/QoE.

3.3 Un plan de contrôle SDN étendu et flexible

3.3.1 Introduction

Comme nous avons eu déjà l'occasion de l'écrire dans l'état de l'art, les architectures SDN basées sur OF sont fortement limitées dans leurs fonctionnalités dans différentes dimensions. Essentiellement, l'approche OF est centralisée avec un contrôleur qui contrôle un grand nombre de SOFs très simples dans le réseau.

Il est par conséquent difficile d'implémenter efficacement et de manière distribuée certaines fonctions usuelles et classiques actuellement utilisées dans les réseaux.

C'est par exemple le cas, des fonctions qui gèrent de grandes quantités de trafic comme les pare-feu, la sécurité ou le Deep Packet Inspection (DPI), ou enfin encore des fonctions qui nécessitent une certaine localité comme le re-routage rapide qui nécessite de réagir presque en temps réel avec des contraintes de temps fortes, ou enfin celles qui nécessitent une analyse intelligente sur les nœuds du réseau comme le monitoring.

Pour rappel l'approche DICES que nous avons présentée dans la section 3.2 vise à répondre à cette problématique en proposant une architecture basée sur différentes fonctions décrites en 3.2.1.

Compte tenu du fait que la définition d'une telle architecture constitue un vaste programme, nous nous sommes intéressés dans le travail à la réalisation d'une preuve de concepts illustrant principalement la faisabilité des items 2 et 3 et aussi l'item 4 de 3.2.1 sous certains aspects.

Ce que nous avons cherché à explorer est la faisabilité d'un plan de contrôle SDN étendu et programmable qui permette de dépasser les limites du protocole OF, comme illustration de principes de l'architecture DICES que nous avons proposée.

L'idée est de pouvoir réaliser des services réseaux intelligents, distribués, et programmables comme ce qu'offre théoriquement OF, mais en dépassant les limites et les inconvénients de cette approche. Nous avons déjà souligné que ces limites sont essentiellement liées à:

- la faible programmabilité du SOF,
- aux contraintes de l'approche centralisée sous-jacente à OF qui impacte le SLA des services et dont découlent différents problèmes (scalabilité, fiabilité...),
- la rigidité du protocole OF imposée par l'approche à base de tables de flots.

Pour cela nous nous sommes fixés le cadre de référence suivant pour notre preuve de concepts:

1. le concept de nœuds de réseau génériques et programmables généralement connus sous le nom de "white box". Ils doivent intégrer un moteur générique d'exécution du service réseau et une bibliothèque de composants élémentaires qui sont les éléments de base de tout service réseau,
2. une logique de décomposition fine des services réseau en composants élémentaires, qui permettent de concevoir des services à la volée correspondant aux demandes clients à l'aide de blocs de construction disponibles sur chaque nœud du réseau,
3. un mécanisme de reconfiguration ou de redéfinition à la volée des services réseaux sur les nœuds génériques sans interruption de service,

4. des agents élémentaires intelligents appelés éléments contrôleurs SDN chargés d'interagir avec le plan de données à différents niveaux de localité.

Ce plan de contrôle SDN est illustré par un Proof of Concept (PoC) implémentant un service de surveillance (monitoring) distribué.

Le service de surveillance peut agir et évoluer de manière différenciée dans le réseau en fonction des besoins liés au trafic et de l'état du réseau.

Le PoC utilise la plate-forme Click décrite dans l'état de l'art [KOHLER et collab., 2000], dans laquelle nous avons développé différents modules [YASSIN et collab., 2017].

Ce PoC aurait pu être aussi développé sur la base d'autres outils comme eBPF ou même VPP qui sont eux aussi adaptés aux environnements virtualisés et qui sont présentés dans l'état de l'art.

3.3.2 Lignes directrices de conception

Deux principes guident la conception de ce PoC que l'environnement Click permet d'illustrer.

1. Tout d'abord, la mise en place de modèles exécutables permettant la définition dynamique de services réseau. Ces modèles sont construits avec des briques de bases du modèle qui décrivent les services réseau élémentaires. Ils sont utilisés et assemblés par l'orchestrateur pour composer, valider et déployer le service visé.

2. Deuxièmement, une fonction générique de contrôleur SDN est disponible et prête à être instanciée et activée dans tous les composants réseaux. Ces fonctions contrôleurs peuvent être organisées de manière hiérarchique à différents niveaux pour gérer la logique du plan de contrôle et les différents niveaux de localité.

La figure 3.8 présente un ensemble de nœuds réseaux génériques virtuels ou physiques communément appelés "white box" qui supportent des services réseaux distribués et différenciés modélisés avec des composants.

Ces composants de services réseau sont assemblés à partir d'un ensemble de fonctions réseaux élémentaires.

Cet assemblage se fait en élaborant un graphe orienté à l'aide d'une approche de type réseaux de Petri comme décrit dans la section précédente.

Ici nous utilisons la modélisation en graphe adoptée par Click qui en est en fait très proche. Il suffit par exemple de considérer un jeton comme un paquet IP.

Un canal de communication achemine les messages de management et de contrôle entre l'orchestrateur et les nœuds du réseau.

Plan de gestion SDN

Le déploiement des composants est assuré à la demande par la fonction d'orchestration via le canal de contrôle et de gestion. Le canal de gestion, représenté par une flèche jaune sur la figure 3.8, permet de configurer de nouveaux composants sur des nœuds correspondants à différentes fonctions du plan de données ou de mettre à jour les composants en cours d'utilisation.

La fonction orchestrateur est en charge des décisions de déploiement des fonctions réseau en fonction des besoins ou du fait de certains processus automatiques. Elle est également en charge du déploiement effectif et de la mise à jour des composants du réseau, en résumé de la gestion de leur cycle de vie.

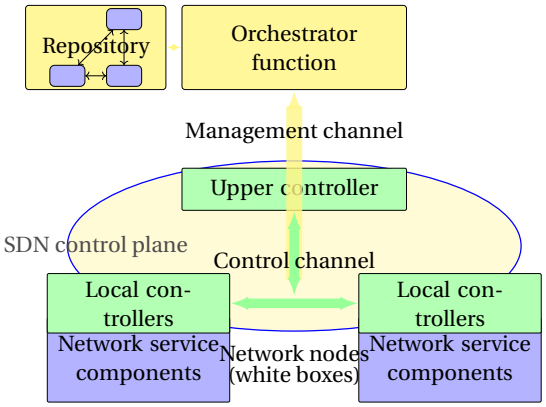


Figure 3.8 – Principes du plan de contrôle & de gestion étendu

Un composant de service réseau peut être entièrement ou partiellement mis à jour, c'est-à-dire que sont mis à jour, supprimés ou remplacés, certains éléments réseau dans le graphe qui modélise le composant.

Plan de contrôle SDN

Le canal du plan de contrôle SDN est encapsulé dans le canal de communication comme le canal de de management. Il est représenté par les flèches vertes qui relient le contrôleur de niveau supérieur et les contrôleurs locaux intégrés dans différents noeuds.

Le plan de contrôle SDN est également réparti entre les contrôleurs de niveau supérieur et les contrôleurs locaux connectés aux éléments qui constituent les composants qui composent le plan de données.

Les contrôleurs supérieurs et les contrôleurs locaux supportent les fonctions intelligentes du plan de données soit localement soit de manière plus globale.

Ces fonctions intelligentes sont réparties entre les contrôleurs locaux et ceux de niveau supérieur pour gérer hiérarchiquement les différents niveaux de localité comme décrits par exemple par [SCHMID et SUOMELA, 2013].

3.3.3 La fonction contrôleur distribuée

Pour implémenter notre fonction de contrôleur SDN distribuée, nous utilisons le concept de contrôleur programmable pour la gestion unifiée des infrastructures réseau virtualisées proposé par [YASSIN et collab., 2017]. Le rôle de ce contrôleur est principalement d'exposer de manière unifiée des ressources hétérogènes aux entités de contrôle et de gestion. Il est également capable de fournir des ressources de calcul pour gérer les décisions de configuration locales ou globales.

Dans le papier de référence, le concept est illustré dans le contexte d'un cas d'utilisation en réseau basé sur l'environnement du routeur logiciel Click de [KOHLER et collab., 2000].

Nous avons décidé de réutiliser cet élément Click contrôleur comme contrôleur SDN dans notre propre conception. Ce contrôleur peut être inséré dans le plan de données et déclenché par l'arrivée de paquets comme d'autres éléments Click. Il peut aussi fonctionner de manière autonome et indépendante de la séquence d'arrivée des paquets. Il est capable de manipuler des gestionnaires d'éléments en lisant et en écrivant des paramètres sur des éléments Click individuels ou sur des ensembles d'éléments comme les composants Click.

Il intègre également un langage syntaxique simple pour décrire les opérations de calcul exécutées à l'aide des paramètres de lecture. Par conséquent, cette fonction de contrôle flexible peut être utilisée pour notre plan de contrôle SDN de manière distribuée ou hiérarchique.

Nous avons étendu cependant ses fonctionnalités en programmation en le dotant de mémoire et l'avons rendu capable de se connecter et d'échanger avec d'autres contrôleurs similaires pour distribuer des fonctions d'analyse et de décision.

3.3.4 Un cas d'usage: supervision distribuée

Motivation du cas d'usage

Les concepts du plan de contrôle SDN étendu proposé sont illustrés par une fonction de supervision distribuée. Les caractéristiques sont décrites ci-dessous en justifient leurs intérêt.

- Une fonction de supervision est un sous-ensemble élémentaire nécessaire à toute fonction de service réseau. Elle doit potentiellement être disponible sur n'importe quel nœud réseau.
- Une fonction de supervision doit pouvoir disposer d'informations structurées disponibles au niveau le plus local. Une telle fonction n'est pas réalisable de manière satisfaisante avec des SOF standards.
- Les fonctions de supervision peuvent être plus ou moins complexes en fonction des besoins de l'opérateur, des scénarios et des contextes . Elles peuvent être structurées en fonction du contexte et mises à niveau dynamiquement, à la demande.
- Une fonction de surveillance peut générer beaucoup de trafic de contrôle, en particulier dans le cas des réseaux SDN OF, où la surveillance est centralisée, sans intelligence de traitement au niveau des nœuds et se fait par pooling.
- Une fonction de supervision doit pouvoir prendre en compte certaines contraintes locales, par exemple si une grande réactivité est requise.
- Une fonction de supervision peut nécessiter beaucoup de ressources lorsqu'une analyse sophistiquée est nécessaire. Elle montre l'intérêt de pouvoir ajuster dynamiquement l'utilisation des ressources informatiques.

Description du cas d'usage

Le scénario du cas d'usage est le suivant. L'objectif est de détecter un flux éléphant, qui consomme trop de bande passante, au détriment des autres flux, puis de prendre des mesures correctives si nécessaire au niveau du nœud qui détecte ce flux, généralement un point d'entrée du réseau. Il s'agit donc d'une action de correction différenciée à un moment donné au niveau d'un nœud du réseau.

Afin d'optimiser les ressources CPU, nous n'intégrons d'abord qu'une fonction de supervision très basique dans les nœuds du réseau. Cette fonction est générique, a peu d'impact sur les ressources et est distribuée de la même manière sur tous les nœuds (étape 1 dans la figure 3.9). Cette fonction de supervision de base ne sait détecter, en fonction du débit sur l'interface, qu'une variation importante et croissante du trafic sur

l'un des nœuds via un micro-contrôleur local et un micro-contrôleur global jouant le rôle de compteur et représenté sur la figure 3.11.

Si une augmentation anormale du trafic est détectée par une fonction contrôleur local, celle-ci envoie une signalisation au contrôleur de niveau supérieur, qui décide alors d'installer, à la volée sur le nœud, une fonction de monitoring plus élaborée, prenant en compte d'autres critères plus fins tels que l'emplacement du nœud dans le graphe du réseau.

Par exemple, s'il s'agit d'un nœud d'entrée, la configuration de supervision plus complexe peut alors effectuer une analyse fine et identifier le débit anormal à l'origine de l'anomalie (étape 2 dans la figure 3.9).

L'identification est réalisée grâce à un service de supervision plus évolué, qui peut identifier les sessions IP et calculer leur débit moyen.

Ce calcul peut être effectué en particulier par des micro-contrôleurs supplémentaires insérés dans la configuration, c'est-à-dire des contrôleurs affectés à la mesure des débits locaux qui interagissent avec le premier contrôleur chargé de la mesure du débit global de la figure 3.12.

Lors de la détection d'une anomalie, une action corrective est mise en place par le contrôleur local (étape 3 dans la figure 3.9). Ici il s'agit d'activer et de configurer un shaper de trafic qui inclut dans la configuration enrichie limitera le débit du flux éléphant.

Un mécanisme d'hystérésis basé sur le volume global du trafic permet de revenir à la configuration de supervision initiale sans risque d'oscillations.

Ce cas d'usage nous permet d'illustrer les propriétés suivantes qui ne peuvent être réalisées dans un contexte OF pour les raisons déjà mentionnées:

- la mise à niveau, sans interruption de service, d'une configuration générique,
- l'optimisation des ressources d'un nœud en réduisant la fonction de supervision au strict nécessaire,
- un traitement intelligent et différencié au niveau du nœud avec une analyse locale des flux,
- la distribution de l'analyse du trafic sur plusieurs contrôleurs élémentaires qui interagissent entre eux et organisent de manière distribuée ou hiérarchique le plan de contrôle,
- la complexification ou la simplification du traitement au niveau d'un nœud, selon les besoins.

3.3.5 Implémentation du cas d'usage

Click

Le routeur virtuel Click développé par [KOHLER et collab., 2000] consiste à exécuter un moteur Click en remplacement partiel ou total des fonctions du réseau noyau Linux standard d'une machine physique ou virtuelle.

Le moteur fournit une bibliothèque d'éléments simples ou complexes qui peuvent être assemblés à l'aide d'un script de configuration pour composer un chemin de données sous la forme d'un graphe orienté, représentant un service particulier.

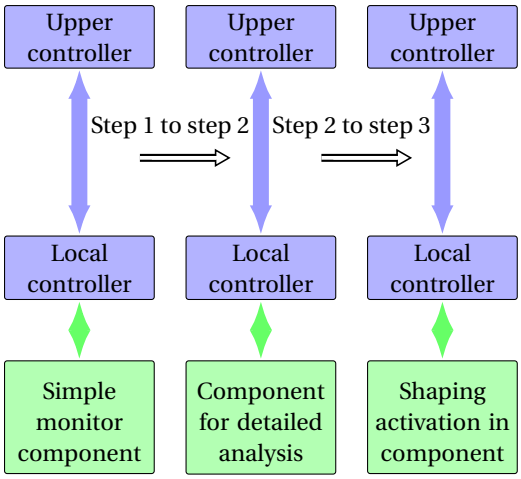


Figure 3.9 – Etapes du cas d'usage de monitoring

Les paquets IP se déplacent le long des arcs orientés entre les sommets qui sont les éléments Clicks et dans lesquels un traitement spécifique est appliqué (classification, réécriture ...). La classe d'un élément spécifie un nombre fixe ou indéfini de ports d'entrée/sortie et son comportement.

La syntaxe générique est la suivante:
$(element_a)[output_x] - packets \rightarrow [input_y](element_b)$.

Les paquets sont traités suivant les fonctions codées dans chaque élément qui sont éventuellement paramétrés par une chaîne de configuration.

A la frontière des composants Click, les éléments *FromDevice* & *ToDevice* réalisent des entrées-sorties de ou vers des interfaces réseau. En outre, les éléments exposent les *handlers* (interfaces) pour échanger entre eux ou avec des *handlers* distants, au moment de l'exécution.

Le moteur de click lui-même expose des *handlers* globaux pour contrôler l'ensemble des opérations. En particulier, l'un d'eux permet de réaliser une reconfiguration à chaud, c'est-à-dire un rechargement rapide à la volée d'un nouveau graphe.

Lors de la conception, le routeur Click est réalisé à partir de sources standards et personnalisées (classes C++), permettant d'étendre la bibliothèque d'éléments.

Click peut être construit pour fonctionner soit en tant qu'application dans l'espace utilisateur, ce qui est plus pratique pour tester/déboguer des développements expérimentaux, soit en tant que module noyau lorsque de meilleures performances sont attendues. Ici nous nous sommes restreints à l'espace utilisateur car notre objectif premier n'était pas les performances.

Un script de configuration est donné en argument lors du lancement du routeur Click. Il comprend la description du graphe et le paramétrage initial de ses éléments.

Fondamentalement, les éléments standard accomplissent des opérations aussi simples que possible et le plus efficacement possible. Un traitement intelligent peut être spécifié par exemple par le biais de règles de classification, pour répartir différents flux sur différents arcs.

La figure 3.10 illustre à titre d'exemple une chaîne de traitement très simple composée de trois éléments et qui compte (élément *counter*) et supprime (élément *discard*) systématiquement les paquets qui arrivent.

Figure 3.10 – Example de chaîne Click simple

Un nouvel élément Click contrôleur

Pour les besoins de notre démonstrateur et pour illustrer les concepts proposés, nous avons étendu les fonctionnalités de l'agent contrôleur Click introduit par [YASSIN et collab., 2017].

Cet élément offre un mécanisme d'automatisation programmable où certaines actions peuvent être définies, grâce à une combinaison logique ou arithmétique de gestionnaires d'éléments globaux ou locaux, suivant une expression de contrôle écrite dans un langage adhoc. Cette expression est évaluée périodiquement, sur événement correspondant à l'arrivée d'un paquet ou sur demande externe.

Si nécessaire, les actions sont lancées en écrivant à des gestionnaires externes arbitraires (par ex. *ExternalCall.active*).

Un élément d'appel externe expose un gestionnaire de commandes informant d'une tâche à effectuer lorsqu'elle est active. Il peut être interrogé par le contrôleur supérieur qui peut à son tour provoquer une rétroaction à distance. Cette logique est utilisée, dans notre expérience pour reconfigurer à la volée le moteur Click.

Pour les besoins de notre démonstrateur et pour illustrer les concepts proposés, nous avons étendu les fonctionnalités de l'agent contrôleur Click introduit par [YASSIN et collab., 2017]. En plus des fonctions logiques de base de composition des prestataires (*handlers*), qui permettent de produire des vues unifiées des composants et des éléments clicks, les points principaux de nos extensions sont :

- La capacité à maintenir des valeurs variables en mémoire. Ces variables permettent, par exemple, de calculer le filtre Infinite Impulse Response (IIR) qui permet de lisser la mesure du trafic et d'effectuer une mesure du débit de trafic.

- La possibilité de lire et d'écrire sur des prestataires d'autres contrôleurs afin de pouvoir composer logiquement ou de manière arithmétique les résultats provenant de plusieurs contrôleurs attachés à différents composants Click.

- La possibilité d'échanger à différents niveaux avec des composants et éléments Click et avec d'autres contrôleurs de manière hiérarchique ou à plat.

Finalement, notre contrôleur peut être programmé pour effectuer les opérations requises par simple configuration comme n'importe quel élément Click.

Configurer un contrôleur consiste à le configurer avec une expression de contrôle qui décrit les opérations à effectuer telles que la lecture ou l'écriture sur des prestataires d'éléments Click, la composition des valeurs de ces prestataires pour produire de nouveaux résultats pour déclencher des actions soit directement au niveau local, soit à un niveau plus global via l'élément *external element* (voir figure 3.11 pour plus de détails).

Description de l'implémentation

La figure 3.11 montre comment la fonction de monitoring de base est implémentée dans Click. Il s'agit de la première étape du processus illustré dans la figure 3.9.

Le flux entrant est récupéré par l'élément Click *From Device* et passe par un élément Click *classifier* qui permet de collecter les flux de données sur lesquels la surveillance s'applique, par exemple les flux TCP ou UDP.

Le flux intéressant passe ensuite par un élément compteur, puis par les éléments d'interface de files d'attente et de sorties.

Un contrôleur local permet de lire le compteur global et de lisser les valeurs extraites. Si la valeur moyenne dépasse un seuil fixe, le contrôleur active l'*external element* qui permet de lever un drapeau. Une fois cet indicateur détecté, le contrôleur central télécharge via le canal de communication une configuration plus sophistiquée de monitoring permettant une analyse détaillée des flux.

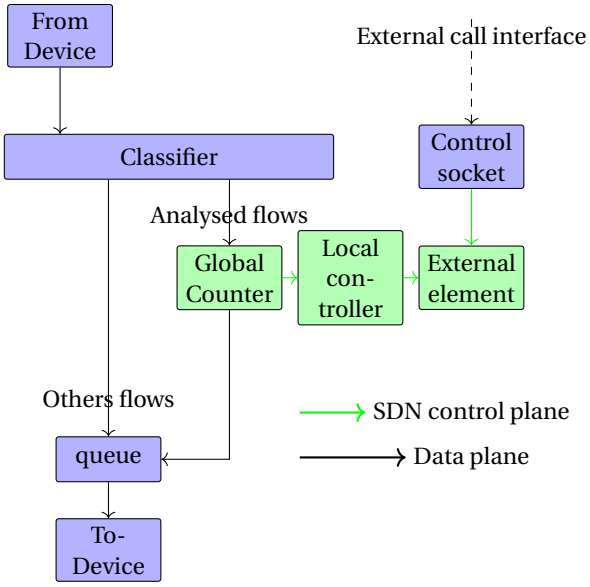


Figure 3.11 – Composant de monitoring Click basique

La figure 3.12 montre comment la fonction de surveillance plus sophistiquée est conçue dans Click, pour permettre une analyse détaillée des flux.

Les éléments jaunes sont ceux qui sont téléchargés et ajoutés dynamiquement à la configuration précédente.

Nous avons développé l'élément *Contrack* qui permet d'identifier et de séparer les différentes sessions TCP ou UDP, sur les différentes sorties.

Il inclut des variables permettant de construire une machine à états finis et l'identification et le suivi des sessions TCP ou UDP. Sur chaque sortie de l'élément Contrack, on trouve une chaîne d'éléments composée d'un élément de comptage, d'un élément de file d'attente et d'un shaper. Cette chaîne de base est utilisée pour mesurer le débit binaire de session sur chaque sortie.

Le contrôleur de débit local de chaque chaîne visible dans le diagramme est utilisé pour mesurer le débit binaire du débit de la session et pour le comparer au débit global mesuré par le contrôleur de débit global.

Les deux contrôleurs travaillent en coordination pour activer le *shaper* si nécessaire par une simple configuration de l'élément *shaper* à l'étape 3 de la figure 3.9. La configuration du *shaper* se fait donc localement.

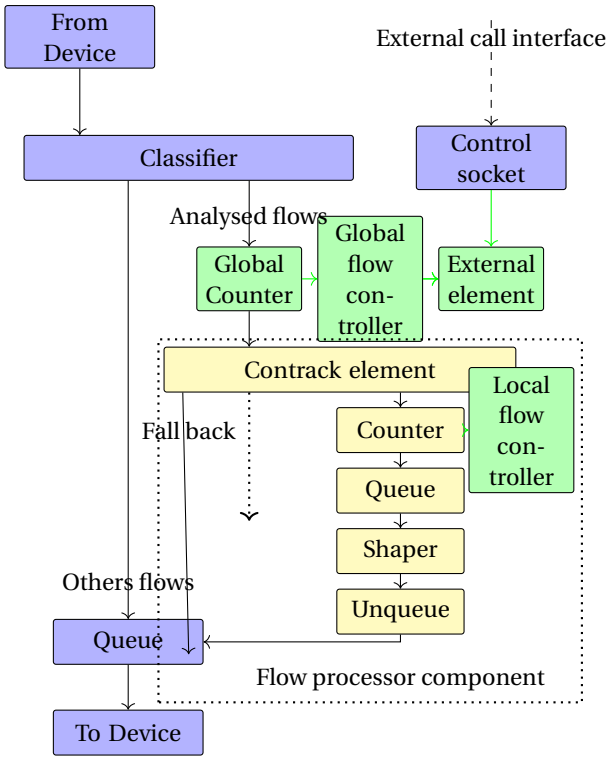


Figure 3.12 – Composant Click de *monitoring* complexe

3.3.6 Résultats et retour d'expérience

La figure 3.13 illustre la preuve de concept composée de trois machines virtuelles fonctionnant sur une machine physique. Le nœud VM2 supporte la fonction de supervision simple ou complexe qui s'applique à un flux routé entre deux interfaces d'entrée-sortie de la machine virtuelle.

Une source de trafic et un récepteur sur les machines virtuelles VM1 et VM3 permettent de réaliser des scénarios de trafic.

Le contrôleur global est installé sur la machine physique et a pour fonction de télécharger la configuration de surpervision complexe. C'est un simple démon sous la forme d'un script Python.

Le générateur de trafic génère successivement dans le temps une séquence de sessions UDP de durées et de débits variables. Quand un flux éléphant est généré, il y a une forte croissance du trafic à un moment donné qui déclenche les étapes 2 et 3 de la figure 3.9.

Une séquence de test est représentée sur la figure 3.14.

Le diagramme montre la progression de la séquence de test que nous avons effectuée avec les flux entrants et sortants. La somme des flux entrants est représentée en bleu, les flux sortants en rouge. Le flux entrant éléphant est en vert, le flux sortant est en orange. On peut voir que le flux éléphant est limité dès son apparition afin de maintenir le trafic global dans un gabarit spécifié.

La transition se fait aux alentours de $t = 50$s. Nous pouvons voir juste à cet instant une brève perte de paquets pendant environ 40ms liée à la mise à jour à la volée du composant de monitoring.

Cette perte montre que la mise à niveau n'est pas totalement transparente et qu'elle a un impact sur le trafic. En effet, dans Click, la mise à niveau à chaud des composants Click se fait par un remplacement complet du composant par le nouveau composant.

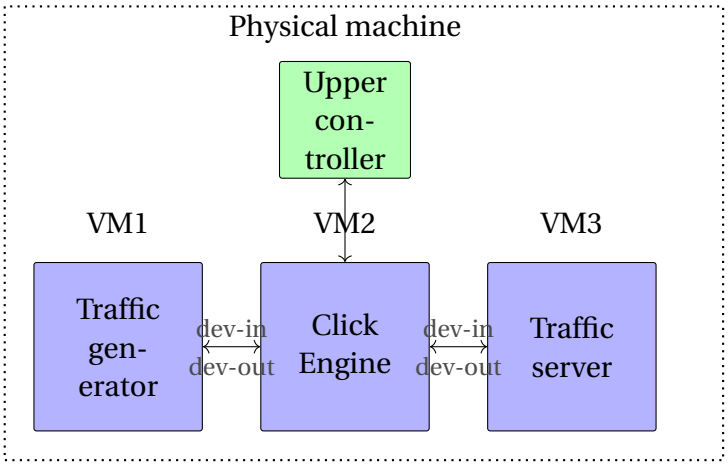


Figure 3.13 – Preuve de concept

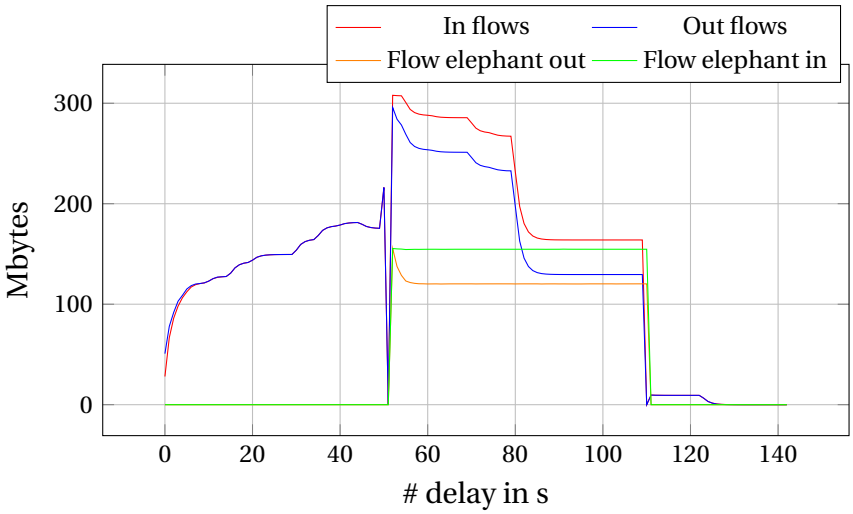


Figure 3.14 – Résultats expérimentaux

Il serait nécessaire de mettre en œuvre des mécanismes de mise à niveau beaucoup plus raffinés agissant séparément sur une partie du graphe des composants Click pour espérer une mise à jour ayant un impact limité. Cela passerait sans doute par une refonte complète de l'architecture de Click. Néanmoins, la perte de paquets reste modérée et relativement brève.

3.4 Conclusion

Dans ce travail, nous avons démontré la faisabilité d'un plan de contrôle SDN étendu et flexible qui surmonte les limites de l'approche OF.

Ces limitations sont les suivantes : Tout d'abord, la faible programmabilité des switchs OF qui empêche de réaliser des traitements locaux sophistiqués.

Deuxièmement, la forte centralisation inhérente à l'architecture OF qui limite les possibilités de distribution des fonctions réseaux dans les éléments réseau.

Nous avons basé ce concept sur plusieurs principes de conception:

- Une approche de type *white box* de switches programmables génériques.

- Une désagrégation des services réseau en composants élémentaires assemblés à l'aide de modèles qui modélisent le service à l'aide d'un modèle de type graphe.

- Un moteur d'exécution des services hébergé sur tous les nœuds

- L'introduction d'une fonction de contrôleur qui prend la forme d'un agent intelligent et programmable qui peut être distribué de manière flexible dans les composants de service réseau.

Nous avons illustré ensuite ce concept sur un cas d'utilisation de supervision adaptative dans le but de détecter et de limiter des flux éléphants.

Nous nous sommes appuyés sur le routeur virtuel Click pour lequel nous avons développé la fonction contrôleur sur la base d'un élément Click.

Cette mise en œuvre nous a permis de démontrer la faisabilité du concept proposé. Elle nous a permis également de mettre en évidence les limitations inhérentes à l'environnement Click.

Tout d'abord, en ce qui concerne l'implémentation actuelle de Click, les chemins de données sont définis de manière statique. Par conséquent, seul un nombre raisonnable de flux peut être pris en compte. Cela peut s'avérer problématique lorsqu'il est nécessaire de gérer des problèmes de routage.

En revanche, la logique à l'intérieur d'un élément donné n'a aucune limite en termes de complexité puisqu'il s'agit de pur logiciel. Par exemple, on peut implémenter une liste ouverte de contextes session-flux, chacune avec une machine d'état. Nous avons développé un tel élément : Contrack implémentant la base logicielle d'un pare-feu *stateful*.

Par conséquent, deux axes apparaissent pour implémenter une logique complexe avec la technologie Click :

- Rassembler toute la logique en éléments multi-fonctions riches et spécifiques, qui peuvent être incompatibles avec l'objectif initial de l'approche Click.

- Distribuer l'algorithme sur plusieurs éléments plus génériques ce qui nécessite de trouver un moyen de garder la connaissance contextuelle entre eux, en utilisant des méta-données par exemple.

En second lieu, il faut souligner les limitations du mécanisme de mise à niveau à chaud des composants Click qui provoque la perte de paquets. Cette mise à jour nécessiterait probablement un mécanisme différencié et affiné de mise à jour des composants afin de réduire les pertes.

Cependant, l'architecture proposée permet d'implémenter, de distribuer ou de centraliser des services réseau flexibles et reconfigurables en fonction des besoins. Elle peut aussi également être implémentée de manière à recréer le scénario primitif OF comme référence. Un composant Click pourrait très bien porter toute la logique d'un contrôleur OF basique.

Dans la continuité de ce travail, il faudrait aussi compléter et spécifier la notion de contrôleur que nous avons présentée en précisant en détail l'étendue des fonctionnalités nécessaires pour assurer une programmabilité et une modularité satisfaisante afin de pouvoir centraliser ou distribuer le plan de contrôle SDN selon les exigences des services réseau, tel que nous l'avons imaginé.

Bibliography

A JAVAN, A. M., M AKHAVAN. 2013, «Simulating turing machines using colored petri nets with priority transitions», *Int. J. on Recent Trends in Engineering and Technology,*, vol. 9, nº 1, p. 7. URL http://searchdl.org/public/journals/2013/IJRTET/9/1/34.pdf.

AFLATOONIAN, A., A. BOUABDALLAH, V. CATROS, K. GUILLOUARD et J.-M. BONNIN. 2014, «An orchestrator-based sdn framework with its northbound interface», dans *Advances in Communication Networking, Lecture Notes in Computer Science (LNCS)*, vol. 8846, Springer International Publishing.

ALI, E. K., M. MANEL et Y. HABIB. 2017, «An efficient mpls-based source routing scheme in software-defined wide area networks (sd-wan)», dans *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, ISSN 2161-5330, p. 1205–1211, doi:10.1109/AICCSA.2017.165.

ANNIE, C.-G. 2006, *Les réseaux de Petri un outil de modélisation*, vol. 1, Sciences Sup, 126-127 p..

BERTHOMIEU, B. et F. VERNADAT. 2006, «Time petri nets analysis with tina», dans *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems*, QEST '06, IEEE Computer Society, Washington, DC, USA, ISBN 0-7695-2665-9, p. 123–124, doi:10.1109/QEST.2006.56. URL http://dx.doi.org/10.1109/QEST.2006.56.

BRUNO, G., A. CASTELLA, R. AGARWAL, I. PAVESIO et M. PESCARMONA. 1994, «Using modular petri nets for developing telecommunication software», dans *Parallel and Distributed Real-Time Systems, 1994. Proceedings of the Second Workshop on*, p. 206–211, doi:10.1109/WPDRTS.1994.365627.

FEHLING, R. 1993, «A concept of hierarchical petri nets with building blocks», dans *Papers from the 12th International Conference on Applications and Theory of Petri Nets: Advances in Petri Nets 1993*, Springer-Verlag, Berlin, Heidelberg, ISBN 3-540-56689-9, p. 148–168. URL http://dl.acm.org/citation.cfm?id=647738.760694.

KNIGHT, S., H. X. NGUYEN, N. FALKNER, R. A. BOWDEN et M. ROUGHAN. 2011, «The internet topology zoo.», *IEEE Journal on Selected Areas in Communications*, vol. 29, nº 9, p. 1765–1775. URL http://dblp.uni-trier.de/db/journals/jsac/jsac29.html#KnightNFBR11.

KOHLER, E., R. MORRIS, B. CHEN, J. JANNOTTI et M. F. KAASHOEK. 2000, «The click modular router», *ACM Trans. Comput. Syst.*, vol. 18, nº 3, doi:10.1145/354871.354874, p. 263–297, ISSN 0734-2071. URL http://doi.acm.org/10.1145/354871.354874.

MORENO, R. et J. SALCEDO. 2008, «Adaptive petri nets implementation. the execution time controller», dans *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, p. 300–307, doi:10.1109/WODES.2008.4605963.

MURATA, T. 1989, «Petri nets: Properties, analysis and applications.», *Proceedings of the IEEE*, vol. 77, nº 4, p. 541–580.

SCHMID, S. et J. SUOMELA. 2013, «Exploiting locality in distributed sdn control», dans *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, ACM, New York, NY, USA, ISBN 978-1-4503-2178-5, p. 121–126, doi:10.1145/2491185.2491198. URL http://doi.acm.org/10.1145/2491185.2491198.

URMILA, R. et U. POL. 2014, «Cloud computing with open source tool :openstack», *American Journal of Engineering Research (AJER)*, vol. 3, p. 233–240.

YASSIN, M., K. GUILLOUARD, M. OUZZIF, R. PICARD et D. ALUZE. 2017, «A programmable controller for unified management of virtualized network infrastructures», dans *2017 IEEE Symposium on Computers and Communications (ISCC)*, p. 1344–1351.

Chapitre 4

Placement de services réseaux dans une infrastructure SDN & NFV

“ One, two! One, two! And through and through The vorpal blade went snicker-snack! He left it dead, and with its head He went galumphing back. ”

Lewis Carroll

Sommaire

4.1 Introduction

Notre travail sur la définition d'une architecture SDN distribuée orientée services réseaux nous a amenés à considérer la problématique du placement de contrôleurs de manière à optimiser le SLA des services déployés.

Nous avons montré sur un exemple simple dans le premier chapitre que le placement optimisé de contrôleurs permettait de respecter une contrainte de latence pour réaliser un service réseau de re-routage rapide.

D'autre part, un contrôleur SDN n'est rien d'autre qu'un logiciel qui s'exécute sur des nœuds physiques ou virtuels et qui consomme des ressources pour réaliser les performances recherchées. Le contrôleur déployé va consommer des ressources au niveau du nœud d'exécution ainsi que de la bande passante dans le réseau. Un des enjeux est qu'il dispose d'un réseau de contrôle satisfaisant.

Le problème que nous avons voulu aborder en premier lieu est le placement optimal de contrôleurs dans une infrastructure pour réaliser les SLA recherchés pour les services réseaux.

Les problèmes d'optimisation qui se posent pour définir des configurations de contrôleurs distribués sont en général NP difficiles. Comme l'état de l'art l'a montré, les approches possibles sont multiples et diverses.

Pour notre part, nous nous sommes intéressés en premier lieu à la faisabilité de la définition d'une heuristique permettant de calculer un placement de contrôleurs de manière très rapide même dans des réseaux comportant plusieurs milliers de nœuds, sur une métrique de latence et une métrique de charge et pour un nombre de contrôleurs non fixés à l'avance.

Nous avons utilisé pour cela un algorithme de classification hiérarchique.

Nous avons ensuite recherché une approche plus générale pouvant éventuellement être étendue à des objectifs d'optimisations multiples et rendue aussi générique que possible. En effet le problème du placement des contrôleurs est généralement multi-objectifs et multi-contraintes.

Nous avons utilisé pour cela les algorithmes évolutionnaires car nous étions intéressés par leur flexibilité et leur généricité.

Avec ce type d'algorithme nous avons défini une métaheuristique d'optimisation de la connectivité des groupements autour des contrôleurs, avec un objectif de charge et un nombre de contrôleurs non connus à l'avance. Nous avons utilisé la métrique de k-connectivité moyenne comme critère à optimiser pour améliorer la fiabilité du réseau SDN et donc des services réseaux associés.

Nous avons cherché ensuite dans le cadre d'une collaboration fructueuse avec M. Quang Pham tran anh qui effectuait un postdoc à B-COM, à étendre la technique des algorithmes évolutionnaires à d'autres problèmes de placement.

Nous nous sommes intéressés aux problèmes de placement de chemins qui sont en fait des problèmes de routage et aux problèmes de placement de chaînes de VNF.

L'objectif était d'essayer de définir un outil générique pour résoudre des problèmes de placement avec la plus grande généricité possible.

4.2 Algorithmes de référence pour le placement

4.2.1 Le problème de couverture par ensembles

Placer des contrôleurs dans un réseau en respectant certaines contraintes ou en optimisant certains objectifs, implique de rattacher des nœuds du réseau à ces contrôleurs. Cela revient à définir une partition du réseau en clusters de nœuds rattachés à chaque contrôleur.

Il y a ici une analogie évidente avec le problème de couverture par ensembles dont nous donnons une description succincte.

Le problème de couverture par ensemble est défini de la manière suivante: On considère une paire (U,S) où U est un ensemble fini d'éléments e_i et S une famille de sous-ensembles de U avec $\cup_{s\in S} s = U$. On définit une application w qui à chaque sous ensemble contenus dans S associe un poids positif: $w : S \rightarrow \mathbb{R}_+$.

Le problème de couverture par ensembles consiste à trouver un ensemble d'éléments (des sous-ensembles) de S dont la somme des poids est minimale et qui couvre tous les éléments de U comme par exemple la réunion des ensembles bleu et noir de la figure 4.1.

Formellement il s'agit de trouver un sous-ensemble couvrant, $R \subseteq S$, vérifiant $\cup_{r\in R} r = U$ tel que $\sum_{r\in R} w(r)$ soit minimal.

Dans le cas du rattachement de nœuds d'un réseau à des contrôleurs, il s'agit d'obtenir une couverture exacte, R est une partition de U. C'est donc une contrainte supplémentaire.

Un problème NP- difficile

Ce problème est connu comme NP- difficile. Une manière de prouver qu'il est NP- difficile consiste à montrer qu'il est réductible à une instance du problème de couverture par sommets dont la version décisionnelle est NP-complète. En se ramenant à un problème déjà connu et caractérisé on démontre le résultat. Le problème de couverture par sommets appartient aux 21 problèmes NP complets identifiés par [KARP, 1972].

Pour rappel, le problème de couverture par sommets consiste à trouver un ensemble minimum de sommets pour couvrir toutes les arêtes d'un graphe.

Par ailleurs, Chvatal [JAIN et collab., 2013] ont montré qu'un algorithme glouton permet d'obtenir une approximation à au plus $1 + ln|U|$ de l'optimal.

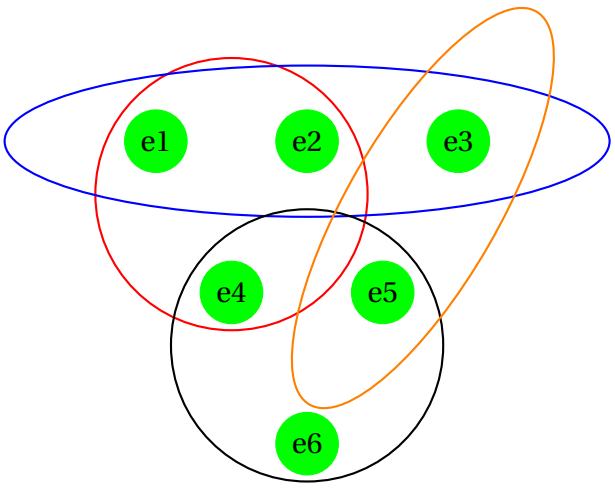


Figure 4.1 – Exemple de couverture par sommets

4.2.2 Le problème de l'emplacement d'installations

Lorsque il s'agit de placer des serveurs de capacités diverses dans un réseau, dans lequel les liens présentent des caractéristiques variées comme la latence, liées par exemple à la topologie géographique dans laquelle est déployé le réseau, apparaît l'analogie avec les problèmes d'emplacement d'installations.

On peut résumer ce type de problèmes en les décrivant comme une situation dans laquelle un certain nombre d'établissements (magasins, hôpitaux,...stations de bases d'un réseau mobile...) désignés sous le terme d'établissements doivent satisfaire un certain nombre de demandes provenant d'utilisateurs (clients, téléphones mobiles...). Dans le cas des serveurs dans un réseau, le problème peut être vu comme discret dans la mesure où les possibilités de localisation des serveurs sont les nœuds du réseau.

Le problème de base est dit sans capacité. Il consiste en considérant d'une part un ensemble de clients et d'autre part un ensemble d'établissements auxquels est associé un coût d'ouverture, à minimiser une fonction objective qui évalue le coût du service rendu. Ce coût dépend généralement de la distance des clients à l'entrepôt auxquels ils sont rattachés d'une part et du coût d'ouverture de chaque entrepôt d'autre part.

Une variante est le problème à capacité "douce". On a alors des entrepôts de capacités limitées en nombre de clients et si le nombre de clients affectés à un entrepôt donné dépasse la capacité de l'entrepôt , il faut ouvrir un nouvel entrepôt autant qu'il sera nécessaire. Le nombre d'entrepôts n'est pas borné.

On peut imaginer de la même manière d'activer de nouveaux contrôleurs dans un réseau ou de nouvelles VNF.

Dans le cas du problème d'emplacement d'établissements avec capacités, le nombre d'entrepôts potentiels est fixé. De plus, à chaque client est associé une demande, et les entrepôts sont dotés d'une capacité limitée pour répondre à cette demande. On cherche dans ce cas à répondre à la demande de chaque client en minimisant le coût de service.

Dans la définition la plus générale du problème, il suffit que la "distance" qui reflète en fait généralement le coût de service soit effectivement une distance au sens d'un espace métrique et vérifie l'inégalité triangulaire pour que l'on puisse parler de problème d'emplacement d'installations.

On voit bien que l'on peut étendre le concept à des problématiques variées comme les problématiques réseau, dans lesquelles la distance considérée n'est plus nécessairement une distance géographique mais par exemple une latence.

A l'inverse un certain nombre de problématiques réseaux font appels à des fonctions de coûts qui ne sont pas nécessairement des distances Dans ce cas une approche de type "emplacement d'établissements" n'est pas nécessairement adaptée.

Un problème NP-difficile

Le problème d'emplacement d'établissements sans capacité est NP-difficile. Une manière de le prouver est de montrer qu'il constitue une cas particulier du problème de couverture par sommets minimum déjà décrit qui est lui même NP-difficile comme décrit dans l'ouvrage de KARP.

4.2.3 K-moyennes, K-médianes , K-centres, K-médoïdes

Si l'on considère que le coût d'ouverture d'un entrepôt est nul, le problème de l'emplacement d'établissements se rapproche des problèmes désignés par K-moyennes, K-médianes et K-centres.

Les différents K correspondent ici à différentes variantes autour de la même idée. Il s'agit d'associer les clients à K entrepôts ou "centres" dont la localisation peut être confondue avec celles de certains clients, de manière optimale en minimisant le coût qui cette fois ne dépend plus que de la distance entre clients et entrepôts et ne dépend pas du nombre d'entrepôts qui est fixé.

Dans la variante K-moyennes on cherche à minimiser la distance moyenne L2 des nœuds au centroïde du groupe qui correspond à un établissement.

Dans K-médianes on minimise la distance moyenne L1.

Dans K-centres on cherche à minimiser la distance maximale L∞ des nœuds à leurs centroïdes.

On désigne la solution du problème par l'ensemble des K groupes trouvés $C_1..C_k$ dotés chacun d'un centroïde $c_1..c_k$, c'est à dire:

$$C_K = \{C_1, \ldots, C_k\} \tag{4.1}$$

Chacun des groupes est constitué à partir des N points qui peuvent être groupés:

$$D_N = \{x_1, \ldots, x_n\} \tag{4.2}$$

Les fonctions objectives à minimiser sont, pour le problème K-moyennes qui se réfère à la distance L_2:

$$J(c_1, \ldots, c_k) = \sum_{j=1}^{k} \sum_{x_i \in C_k} ||x_i - c_k||^2 \tag{4.3}$$

Pour le problème K-médianes qui se réfère à la distance L_1:

$$J(c_1, \ldots, c_k) = \sum_{j=1}^{k} \sum_{x_i \in c_k} |x_i - c_k| \tag{4.4}$$

et pour le problème K-centres on préfère prendre le pire cas et minimiser la distance maximale aux différents centroïdes:

$$J(c_1, \ldots, c_k) = \max_{j=1}^{k} \min_{x_i \in c_k} d(x_i, c_k) \tag{4.5}$$

La notion de médoïde signifie simplement qu'on considère un point existant d'un ensemble de points pour le centroïde plutôt qu'un point artificiel de type barycentre d'un ensemble de points. C'est une notion qui convient bien à la problématique du placement des contrôleurs.

4.2.4 Techniques de regroupement (clustering) pour le placement

Placer des contrôleurs dans un réseau peut être vu comme équivalent à regrouper des nœuds du réseau autour de contrôleurs logiques ou physiques. L'idée vient naturellement d'aborder le problème en s'appuyant sur des techniques de regroupement qui sont couramment utilisées dans le traitement des grandes quantités de données.

Regrouper signifie regrouper des données en classes, d'une manière similaire à ce qui est recherché dans l'apprentissage ou la classification non supervisée.

La définition du terme groupe (*cluster*) reste cependant floue et désigne le fait de rassembler des objets dans des classes sur des critères de similarités entre objets et de différences entre classes. On utilise aussi parfois le terme de partitionnement en particulier pour désigner l'application de techniques de regroupement aux graphes.

Deux approches principales permettent de regrouper, la notion de similarité ou la notion de densité.

Groupes basés sur la similarité On utilise une métrique caractérisant une similarité et la similarité entre les éléments du groupe est plus grande qu'entre des éléments quelconques de groupes différents. Ce peut-être par exemple la distance euclidienne ou encore le nombre de sauts, ou une distance L^n qui sont des exemples de mesures de similarités évidentes.

Regroupement basés sur une notion de densité Une manière de définir des groupes peut être de distinguer des zones à forte densité et d'autres à faible densité dans un ensemble de données. Ce peut être une approche efficace par exemple quand les frontières entre groupes sont floues ou bruitées. Dans le cas d'un graphe, on peut utiliser la notion de densité d'un graphe qui est définie comme le rapport du nombre d'arcs du graphe (ou d'un sous-graphe) au nombre total possible d'arcs du graphe (sous-graphe) complet qui vaut $\frac{N\cdot(N-1)}{2}$. D'autres typologies existent mais qui n'ont pas d'intérêt dans le cadre de ce rapport.

Regroupement dans des graphes Cet aspect nous concerne directement puisque les réseaux sont modélisés par des graphes.

On peut en premier lieu remarquer qu'un problème de regroupement de points d'un graphique peut être modélisé sous forme de graphe. Il suffit de construire un graphe complet où chaque nœud correspond à un point géométrique et les arcs entre les points sont pondérés avec les distances entre les différents points.

On considère ici plus généralement des données qui peuvent être représentées par un graphe avec les arcs du graphe représentant des connexions entre les nœuds du graphe. Par exemple des objets graphiques peuvent être considérés comme connectés s'ils sont suffisamment proches et on retrouve la notion de similitude. La structure de graphe est compatible avec de nombreux algorithmes de regroupement basés sur une notion de similarité s'appuyant sur une notion de distance.

Un des points clés du regroupement dans les graphes est la notion de matrice de similarité. Une matrice de similarité très souvent utilisée dans les graphes est la matrice des plus courts chemins mais d'autres sont possibles qui cherchent à représenter différentes propriétés du graphe comme la connectivité, la *betwenness centrality* [1] [BRANDES, 2001] ...

Une autre manière de définir un groupe peut être d'identifier des composantes fortement connectées du graphe. A l'extrême, on peut chercher à identifier les sous-graphes complètement connectés ou les cliques.

L'intérêt principal de la modélisation sous forme de graphes est qu'elle permet de représenter une grande variété de problèmes. On peut alors s'appuyer sur la structure du graphe et ses propriétés pour construire les groupes et appliquer des algorithmes.

Dans le cas d'un graphe modélisant un réseau de télécommunication, on est amené à pondérer les arcs du graphe et les nœuds du graphe avec des propriétés telles que bande passante, taux de pertes, latence, capacité mémoire, CPU, capacité de stockage.

Une distance qui est alors naturellement utilisée est la distance au sens du plus court chemin entre deux nœuds. Bien que ce ne soit pas la distance euclidienne, elle vérifie les

[1] nombre de plus courts chemins passant par un nœud ou un arc du graphe

axiomes de base d'une distance dans un graphe G(V, E):

$$\forall (a,b) \in \mathrm{V}^2,\ d(a,b) = d(b,a)$$

$$\forall (a,b) \in \mathrm{V}^2,\ d(a,b) = 0 \Leftrightarrow a = b$$

Et surtout l'inégalité triangulaire:

$$\forall (a,b,c) \in \mathrm{V}^3,\ d(a,c) \leq d(a,b) + d(b,c)$$

4.3 Regroupement hiérarchique basé sur la latence et la charge

Parmi les techniques de base de regroupement, deux types d'algorithmes sont couramment utilisés, les algorithmes de type k-moyennes et les algorithmes de regroupement hiérarchique.

Nous nous sommes intéressés à la technique de regroupement hiérarchique pour construire une heuristique de placement des contrôleurs pour les raisons suivantes:

1. La démarche du regroupement hiérarchique est intuitive et simple et son implémentation est relativement directe,

2. Nous cherchions une approche dans laquelle le nombre de contrôleurs n'aurait pas à être fixé préalablement, au contraire de K-moyennes,

3. Nous cherchions une approche dans laquelle le temps de calcul ne soit pas prohibitif même pour des réseaux de plusieurs milliers de nœuds,

4. Nous cherchions une approche dans laquelle il soit possible d'introduire des objectifs complémentaires comme l'amélioration de l'équilibre de charge entre les contrôleurs,

5. Nous cherchions une alternative à K-moyennes qui a fait l'objet de différentes publications, et dont la qualité des résultats du point de vue de l'optimisation ne peut être garantie bien qu'en pratique ils soient généralement excellents.

4.3.1 Objectif et problème traité

Notre objectif a été dans cette contribution, de rechercher une heuristique alternative à un algorithme dérivé de l'heuristique K-moyennes qui est présenté dans certains articles sur le placement de contrôleurs.

Nous voulions que notre algorithme permette de minimiser autant que possible le nombre de contrôleurs en respectant un critère de latence à ne pas dépasser pour des services réseaux dont le plan de contrôle est centralisé.

L'heuristique devait aussi respecter un critère d'équilibre de charge entre les groupes attachés à chaque contrôleur.

Le respect d'une contrainte de latence et d'une contrainte de charge sont des éléments permettant de garantir la qualité des services réseaux déployés dans le réseau. Un contrôleur trop chargé ne pourra pas répondre à temps à des sollicitations, du fait de ses capacités de traitement limitées, de la saturation de ses files d'attentes ou du goulot d'étranglement provoqué par le trafic de contrôle.

Un contrôleur trop éloigné des SOFs contrôlés ne pourra pas réagir en temps voulu à des sollicitations locales pour déployer des services réseaux qui impliquent des contraintes temps réels.

Un autre point que nous voulions prendre en compte était la possibilité d'étendre de manière flexible l'heuristique à d'autres objectifs à optimiser. La structure de l'algorithme le permet.

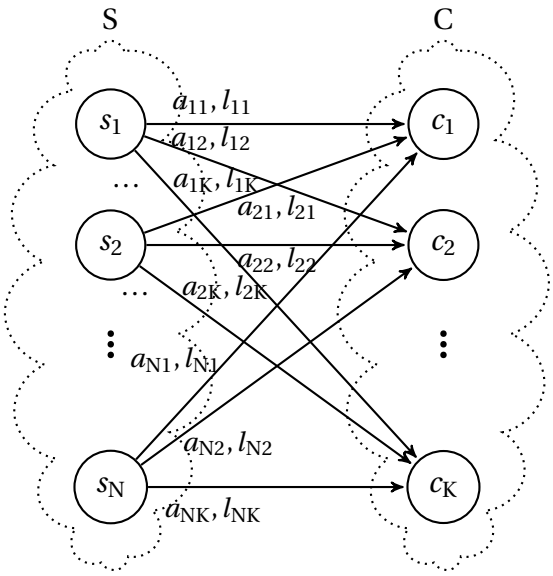


Figure 4.2 – Modèle du système

4.3.2 Modèle ILP de référence

Nous détaillons ici le modèle de référence adopté qui nous a permis d'effectuer des comparaisons avec les résultats obtenus avec un solveur MILP.

La description s'appuie sur le schéma en figure 4.2.

Le modèle que nous avons défini représente l'attachement des SOFs aux contrôleurs. Il doit aussi prendre en compte des facteurs comme la latence des liens et la charge induite par les SOFs dans les contrôleurs.

Nous avons utilisé un graphe bipartite avec G(S, C, E), où S représente l'ensemble des SOFs d'un coté et C l'ensemble des contrôleurs potentiels de l'autre côté. Enfin E est l'ensemble des liens logiques qui connectent les SOFs aux contrôleurs.

Dans notre modèle nous considérons seulement comme connexions possibles les plus courts chemins qui connectent les SOFs aux contrôleurs. Par conséquent, leur latence est simplement la somme des latences induite par les liens et les nœuds qui composent le chemin.

Des contributions supplémentaires à la latence peuvent intervenir au niveau des SOFs et des contrôleurs du fait des traitements applicatifs et du remplissage des files d'attentes. Celles-ci complexifieraient beaucoup le modèle du fait de leur aspect dynamique.

Nous listons ci-dessous les différents paramètres et variables du modèle:

- s_i représente un SOF d'indice i, avec i un entier dans l'intervalle [1, N].
- c_j: représente un contrôleur d'indice j, avec j un entier dans l'intervalle [1, K].
- a_{ij}: représente l'attachement d'un SOF i à un contrôleur j. La valeur de a_{ij} vaut 1, si le SOF i est attaché au contrôleur j. Sinon a_{ij} vaut 0.
- l_{ij}: représente la latence du plus court chemin entre le SOF i et le contrôleur j.

Dans la suite nous considérerons qu'un SOF particulier ne peut être attaché qu'à un contrôleur dans un intervalle de temps donné. Par conséquent nous avons la contrainte suivante:

$$\sum_{j=1}^{K} a_{ij} = 1, \text{pour tout } i \tag{4.6}$$

Où:

$$a_{ij} = \begin{cases} 1 & \textbf{si } s_i \text{ est attaché à } c_j \\ 0 & \textbf{autrement}. \end{cases}, \quad \text{pour tout } i,\ j.$$

Il faut noter que le nombre de contrôleurs K est plus petit ou égal au nombre de SOFs (i.e. K ≤ N).

4.3.3 Modélisation du problème d'optimisation

L'objectif de l'approche proposée, qui vise à minimiser le nombre de contrôleurs SDNs tout en optimisant leur placement, est double :

- minimiser la latence entre les SOFs et leur contrôleur afin d'améliorer la réactivité du système. La réactivité du réseau est une caractéristique clé qui doit être prise en compte, comme par exemple dans le cas d'un service qui vise à rétablir des liens défectueux,
- équilibrer la charge des contrôleurs.

Pour un nombre de contrôleurs K fixé, nous construisons la fonction objective suivante J:

$$\mathrm{J}(a_{11}, \ldots, a_{\mathrm{NK}}) = \sum_{i=1}^{\mathrm{N}} \sum_{j=1}^{\mathrm{K}} a_{ij}\, l_{ij} \tag{4.7}$$

Le problème d'optimisation s'écrit alors:

$$\begin{aligned} \underset{a_{11},\ldots,a_{\mathrm{NK}}}{\text{minimize}} \quad & \mathrm{J}(a_{11}, \ldots, a_{\mathrm{NK}}) \\ & s.t.\ (4.6). \end{aligned} \tag{4.8}$$

Problème dual

Le problème de regroupement exprimé en (4.8) ne minimise pas le nombre de contrôleurs (puisque K est fixé) mais seulement la latence entre les switches et les contrôleurs correspondants. Dans la suite nous exprimons le problème dual qui vise à minimiser le nombre de contrôleurs tout en intégrant l'objectif exprimé en (4.7) comme une contrainte. Par conséquent la fonction objective duale J′ est donnée par:

$$\mathrm{J}'(a_{11}, \ldots, a_{\mathrm{NK}}, \mathrm{K}) = \mathrm{K} \tag{4.9}$$

Et le nouveau problème d'optimisation s'écrit:

$$\begin{aligned} \underset{a_{11},\ldots,a_{\mathrm{NK}},\mathrm{K}}{\text{minimize}} \quad & \mathrm{J}'(a_{11}, \ldots, a_{\mathrm{NK}}, \mathrm{K}) \\ \text{subject to} \quad & (4.6), (4.11) \text{ and } (4.12). \end{aligned} \tag{4.10}$$

$$l_{ij} \leq l_{max}, \text{for all } i,\ j \tag{4.11}$$

$$\max_{j=1,\ldots,\mathrm{K}} \left\{ \sum_{i=1}^{\mathrm{N}} a_{ij} \right\} - \min_{j=1,\ldots,\mathrm{K}} \left\{ \sum_{i=1}^{\mathrm{N}} a_{ij} \right\} < \epsilon \tag{4.12}$$

Où l_{max} est la latence maximum autorisée entre le contrôleur et les SOFs qui lui sont rattachés de manière à tenir les contraintes propres au service réseau.

Le paramètre ϵ représente la différence maximum autorisée entre la charge des différents contrôleurs, c'est à dire la différence en nombre de SOFs dans la mesure où nous considérons chacun des SOFs avec la même charge. Le paramètre ϵ permet ainsi d'obtenir une charge équilibrée entre contrôleurs. Si la valeur de ϵ est trop petite, il y a cependant le risque que le problème ne soit pas résolvable.

4.3.4 Complexité du problème

Problème de référence

Le problème de couverture par ensembles minimum est défini comme suit.

Soit (U, P) où U et un ensemble fini et P une famille de sous-ensembles (une partition de U avec ∪P = U).

Soit une fonction f de P à valeurs dans $\mathbb{R}+$. Nous voulons trouver un sous-ensemble R de P avec $\cup_{p\in R} p = U$ et minimiser $\sum_{p\in R} f(p)$. Le problème de reconnaissance associé à ce problème est connu pour être NP complet (KARP [2010]).

En définissant U = S, et en définissant la fonction f comme une fonction de la distance entre les nœuds de chaque groupe et le centroïde du groupe correspondant (c'est-à-dire les contrôleurs), nous pouvons attacher un poids w aux groupes. Cela prouve que notre problème de référence est un cas particulier du problème de couverture par ensembles décrit en 4.2.1. Par conséquent, la version de reconnaissance de notre problème de référence s'avère être NP-complet.

Le facteur d'approximation α d'un algorithme polynomial donne une borne de la qualité de la solution qui peut être produite par l'algorithme par rapport à la solution optimale fournie en force brute.

Un facteur d'approximation de $\alpha \simeq 2$ est donné par [BAR-ILAN et PELEG, 1991b] pour le problème K-centres non géométrique (la distance considérée n'est pas euclidienne), ce qui correspond à notre problème de référence.

Problème dual

Le problème de l'ensemble dominant est défini comme suit. Un ensemble dominant pour le graphe G = (V, E) est un sous-ensemble D de V tel que chaque sommet sur D est adjacent à au moins un membre de V. La version de reconnaissance du problème de minimiser |D| est connu pour être NP-complet d'après [GAREY et JOHNSON, 1990].

En posant D = C où C est l'ensemble des contrôleurs, et en définissant la relation de proximité comme $l_{ij} \leq l_{max}$, nous prouvons que le problème dual est un cas particulier du problème de l'ensemble dominant. Par conséquent, notre problème dual ou plutôt sa version de reconnaissance s'avère lui aussi être NP-complet.

Un facteur d'approximation de log(|S|) + 1 est donné par [BAR-ILAN et PELEG, 1991b] pour le problème dual.

4.3.5 Deux techniques de regroupement pour le placement de contrôleurs

Nous présentons, dans ce qui suit, deux algorithmes de regroupement avec l'objectif de minimiser le nombre de groupes tout en vérifiant le critère de latence, tel qu'il est exprimé en 4.3.2.

Adapted K-mean clustering

Le premier algorithme, appelé Adapted K-mean est une variation de la technique classique de regroupement K-moyennes, dans laquelle le nombre de groupes est déterminé dynamiquement.

Algorithm 1: Adapted K-mean clustering

input : S, C, l_{max}, (l_{ij}) for all i, j
output: *ClustersList*

```
1  main()
2  ClustersList ← {};
3  curCtrl ← getRandom(C);
4  while (S ≠ {}) do
5      curCluster ← getCluster(curCtrl) ;
6      bestCtrl ← getBestCtrl(curCluster);
7      if (curCtrl ≠ bestCtrl) then
8          curCluster ← getCluster(bestCtrl);
9      ClustersList ← ClustersList ∪ curCluster;
10     S ← S − curCluster;
11     curCtrl ← getFurthest(curCtrl);
12 return ClustersList;

13 getCluster(Ctrl)
14 j ← index(Ctrl) ;
15 for i = 1 to N do
16     if (l_ij ≤ l_max) then
17         curCluster ← curCluster ∪ S_i;
18 return curCluster ;
```

L'idée principale est de commencer par sélectionner aléatoirement un contrôleur (à l'aide de la fonction `getRandom()`) et d'y attacher temporairement tous les SOFs respectant le critère de latence l_{max}. Le centroïde de ce groupe est ensuite déterminé (à l'aide de la fonction `getBestCtrl()`), et le contrôleur le plus proche est sélectionné pour l'attachement final des SOFs, ce qui peut nécessiter l'attachement ou le détachement de certains d'entre eux. Après cela, un nouveau contrôleur, le plus éloigné est sélectionné (en utilisant la fonction `getFurthest()`) pour construire le grouper suivant (plutôt que de manière aléatoire). Ce processus est répété jusqu'à ce que tous les SOFs soient dans un groupe.

L'algorithme adapted K-mean proposé ne permet pas d'homogénéiser la charge des groupes car le nombre de contrôleurs K ne peut pas être connu à l'avance. Par conséquent, il est difficile de limiter la taille des groupes lorsque les différents SOFs sont intégrés dans les groupes.

Une solution pourrait être cependant d'estimer K en calculant le ratio $\frac{d}{l_{max}}$, où d est le diamètre du graphe calculé sur la métrique de latence. Mais cette estimation ne garantirait pas une efficacité quelconque du fait que les topologies des réseaux sont fortement hétérogènes.

Nous pourrions par exemple avoir des nœuds très isolés ce qui empêcherait définitivement l'algorithme d'homogénéiser la charge des groupes. Il faut remarquer que la

complexité de l'algorithme est bornée en $O(N^2)$ car l'algorithme est appliqué pour chaque contrôleur, et la construction des groupes est limitée par le nombre de SOFs borné en O(N).

Cependant, l'algorithme peut être répété plusieurs fois en initialisant chaque fois sur un nœud différent, par exemple m fois, de manière à essayer de trouver la meilleure solution possible en terme de compromis entre le nombre de groupes et le déséquilibre de charge entre les groupes). Cela augmente sa complexité en $O(m \times N^2)$, ou en $O(N^3)$ si on réalise une itération de l'algorithme initiée sur tous les nœuds du réseau.

C'est l'approche que nous avons adoptée compte tenu de la taille des topologies réseaux que nous avons manipulées.

Regroupement hiérarchique

Le deuxième algorithme utilise une technique de regroupement hiérarchique pour l'attachement des SOF aux contrôleurs. L'un de ses objectifs est d'équilibrer la charge des différents groupes.

L'idée principale est de considérer chaque nœud du réseau comme un groupe composé d'un seul SOF et de les fusionner (à l'aide de la fonction `merge`) en utilisant progressivement un critère de proximité.

Deux groupes ne sont fusionnés que lorsque la distance entre le centroïde résultant de la fusion et tous les SOFs du groupe résultant (en utilisant la procédure `maxLatency(.)`) est inférieure à l_{max}.

Pour homogénéiser la charge des groupes résultants, deux groupes ne sont fusionnés que lorsque la taille du groupe résultant (en utilisant la fonction `Size(.)`) est inférieure ou égale à un paramètre qui évolue dynamiquement avec le processus de fusion.

Ce paramètre est déterminé par la différence relative entre les groupes les plus chargés (en utilisant `maxSize(.)` et les moins chargés (en utilisant `minSize(.)`), et en fonction de la valeur de deux coefficients empiriques α et β. Les valeurs de α doivent être supérieures ou égales à 0.5 afin de permettre la fusion des groupes au début de l'algorithme.

Comme α représente une borne inférieure, elle doit être inférieure ou égale à β. Par conséquent, lorsque les groupes sont suffisamment proches et suffisamment petits, ils peuvent être fusionnés. Ceci permet de limiter le nombre de groupes sans violer les contraintes de latence et de charge.

Il faut remarquer que la stratégie principale de l'algorithme est d'éviter de fusionner de grands groupes. Cependant, lorsque le déséquilibre de charge des groupes est modéré, l'agrégation des groupes est autorisée afin de réduire le nombre de contrôleurs résultants.

L'un des principaux avantages de l'algorithme est sa complexité réduite et ses possibilités d'extension. En effet, l'agrégation des groupes peut facilement intégrer plus de contraintes que celles considérées dans l'algorithme 2. On pourrait par exemple intégrer un critère de fiabilité des groupes.

La complexité de l'algorithme est bornée en $O(N^3)$ et est plus élevée que l'algorithme K-mean adapté. Cependant, dans les deux cas, cette complexité reste acceptable pour les topologies de réseau considérées en termes de nombre de nœuds.

D'autre part, une approche en force brute ne peut pas fournir une solution acceptable en un temps raisonnable étant donné la taille des réseaux considérés.

Remarques sur la complexité des deux algorithmes

La complexité des deux algorithmes reste compatible avec une contrainte de temps de calcul inférieure à quelques secondes pour réaliser le placement des contrôleurs dans

Algorithm 2: Hierarchical clustering

input : S, C, l_{max}, α, β, (l_{ij}) for all i, j
output: *ClustersList*

```
1  main()
2  canMerge ← true ;
3  ClustersList ← S ;
4  while canMerge do
5  |   maxLoad ← getMaxLoad(ClustersList) ;
6  |   canMerge ← mergeClusters(ClustersList, maxLoad) ;
7  return ClustersList;

8  getMaxLoad(Clusters)
9  diff ← (maxSize(Clusters) − minSize(Clusters)) / maxSize(Clusters);
10 if (diff ≤ α) then
11 |   maxLoad ← 2 ∗ maxSize(Clusters);
12 else if diff ≥ β then
13 |   maxLoad ← 2 ∗ minSize(Clusters);
14 else
15 |   maxLoad ← 2 ∗ averageSize(Clusters);
16 return maxLoad;

17 mergeClusters(ClustersList, maxLoad)
18 modified ← false ;
19 forall (c_i, c_j) in ClustersList do
20 |   c ← c_i ∪ c_j ;
21 |   if (Size(c) ≤ maxLoad) then
22 |   |   if (maxLatency(c) < l_max) then
23 |   |   |   merge(c_i, c_j);
24 |   |   |   modified ← true ;
25 return modified ;
```

des réseaux de taille moyenne (quelques centaines de nœuds).

Il faut noter que notre implémentation a été réalisée en Python et qu'un codage optimisé permettrait un gain d'un facteur 10 à 100. Remarquons aussi que les deux algorithmes utilisent en entrée la matrice de plus court chemin du réseau. Cette matrice est calculée à l'aide d'un algorithme Floyd-Warshall dont la complexité est en $O(N^3)$. Cependant, ce calcul peut être effectué en différé, et la matrice est dans ce cas une des données en entrée de l'algorithme.

Dans la partie évaluation, nous montrons que l'exécution répétée de l'algorithme permet d'améliorer considérablement les résultats obtenus si de l'aléatoire est introduit en mélangeant aléatoirement m fois l'ordre de la liste initiale des clusters. Dans ce cas, la complexité de l'algorithme peut augmenter jusqu'à $O(m \times N^3)$, où m est le nombre d'essais aléatoires.

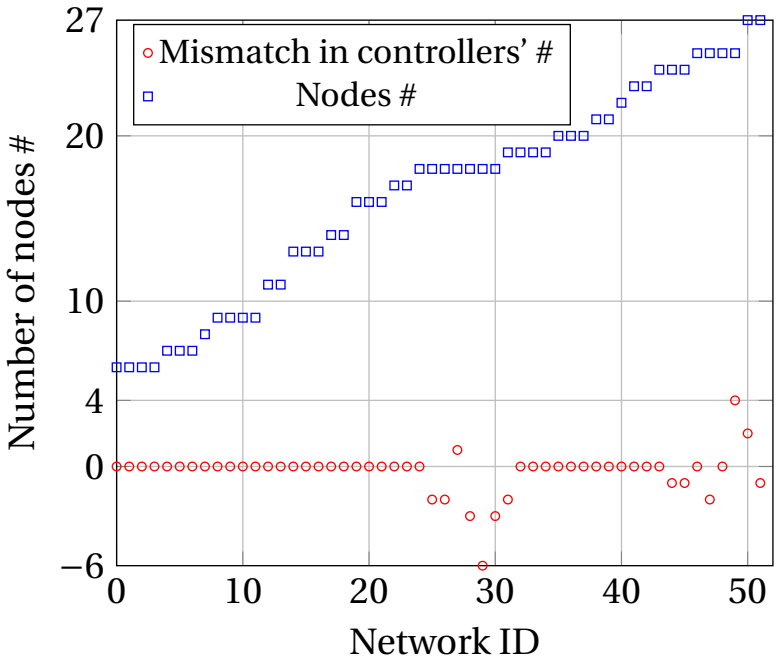


Figure 4.3 – Décalage entre la solution optimale et l'heuristique hiérarchique après 3h de calcul du solveur

4.3.6 Qualification par comparaison avec un solveur

Afin d'avoir une solution de référence, nous avons comparé l'approche hiérarchique proposée avec les résultats optimaux obtenus en utilisant le solveur Gecod [TACK, 2009] et le langage de modélisation minizinc proposé par [STUCKEY et collab., 2014] pour exprimer notre modèle présenté en 4.3.2.

La figure 4.3 représente le décalage sur la charge obtenue sur les clusters en utilisant la solution proposée et la solution optimale basée sur le modèle.

Le temps d'exécution prohibitif nécessaire pour produire des solutions avec le solveur nous a limité aux réseaux avec un nombre de nœuds inférieur à 30. Les résultats obtenus sont cependant la plupart du temps identiques ou très proches de ceux fournis par l'algorithme. On remarque que certains des résultats obtenus en utilisant l'approche du solveur optimal sont plus mauvais (valeurs négatives) que les résultats obtenus avec notre algorithme. Cela s'explique par le temps limité planifié pour résoudre le problème avec le solveur pour un réseau (3 heures).

4.3.7 Qualification sur des topologies typiques

Le comportement de l'algorithme a été qualifié sur des topologies typiques, en posant des hypothèses de latence. La topologie 4.4 représente deux étoiles connectées par un lien. La contribution à la charge du plan de contrôle est normalisée à 1 pour tous les SOF. La charge correspond donc simplement au nombre de SOFs attachés au contrôleur. La latence entre les SOF 4 et 5 vaut 20 ms tandis que la latence entre les autres liens est deux fois moins importante: 10 ms.

La contrainte de latence est fixée dans ce scénario à 15 ms. Dans ce cas l'algorithme retourne les nœuds 4 et 5 comme contrôleurs, chacun étant rattaché respectivement aux clusters $\{1,2,3,4\}$ et $\{5,6,7,8\}$.

Si on réduit la contrainte latence au dessous de 10 ms, chaque nœud devient con-

trôleur. Si on l'augmente au delà du seuil de 40 ms, un des nœuds les plus central 4 ou 5, devient le contrôleur unique comme dans la figure 4.5.

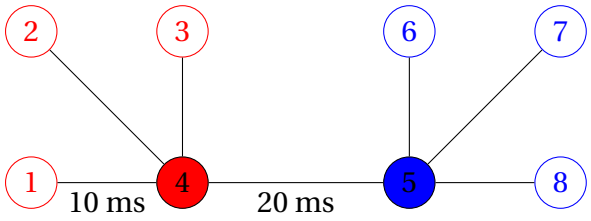


Figure 4.4 – Exemple de réseau en étoile: 2 contrôleurs

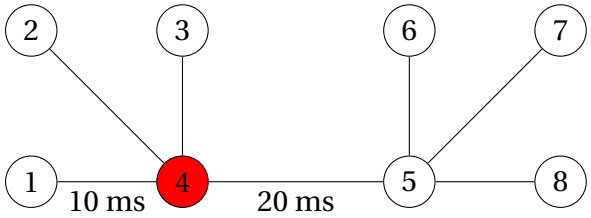


Figure 4.5 – Exemple de réseau en étoile: 1 contrôleur

La topologie de la figure 4.6 est une simple ligne de 8 nœuds réseau séparés par une latence de 20 ms. Dans ce cas, si l'on fixe la contrainte de latence à 20 ms, on obtient, les nœuds 1, 4 et 7 comme contrôleurs, ce qui vérifie effectivement les contraintes. Si la contrainte de latence devient inférieure à 20 ms tous les nœuds deviennent contrôleurs.

Si on fixe la contrainte de latence à 60 ms, les nœuds 2 et 6 deviennent contrôleurs avec deux clusters parfaitement équilibrés $\{1,2,3,4\}$ et $\{5,6,7,8\}$, ce qui est permis par le facteur d'équilibre de charge que prend en compte l'algorithme. Si on fixe la contrainte de latence à 80 ms, l'algorithme sélectionne un seul contrôleur sur le nœud 3.

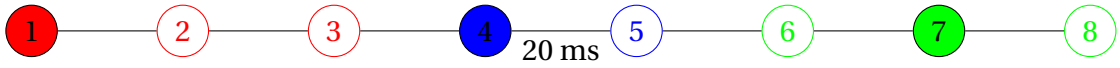


Figure 4.6 – Exemple de réseau en ligne

Nous examinons aussi le comportement de l'algorithme sur une topologie en bus comme dans la figure 4.7. Dans ce cas, si nous fixons la contrainte de latence à 20 ms, les nœuds $1,3,5,7$ deviennent contrôleurs, conformément à l'intuition.

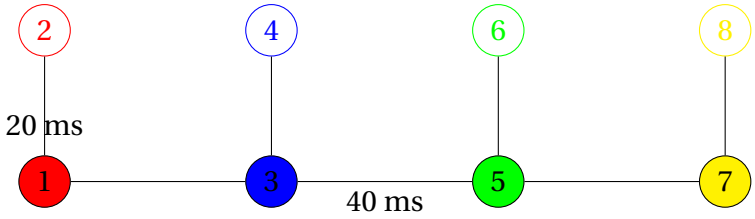


Figure 4.7 – Exemple de réseau en bus

Enfin nous avons essayé d'évaluer le comportement de l'algorithme sur une topologie simple inspirée de celle d'un réseau de télécommunication avec un réseau cœur, un réseau d'agrégation et un réseau d'accès, tous trois fortement maillés. Nous avons fixé arbitrairement la latence des liens dans le réseau cœur à 50 ms, celle du réseau d'agrégation

à 30 ms et celle du réseau d'accès à 10 ms. Comme le réseau est très maillé, la latence la plus grande entre deux nœuds est de 90 ms. En fixant la contrainte de latence à 45 ms, nous obtenons dans ce cas trois groupes organisés autour des SOF du réseau d'agrégation avec les nœuds 4, 13, 10 qui jouent le rôle de contrôleurs, comme illustré dans la figure 4.8.

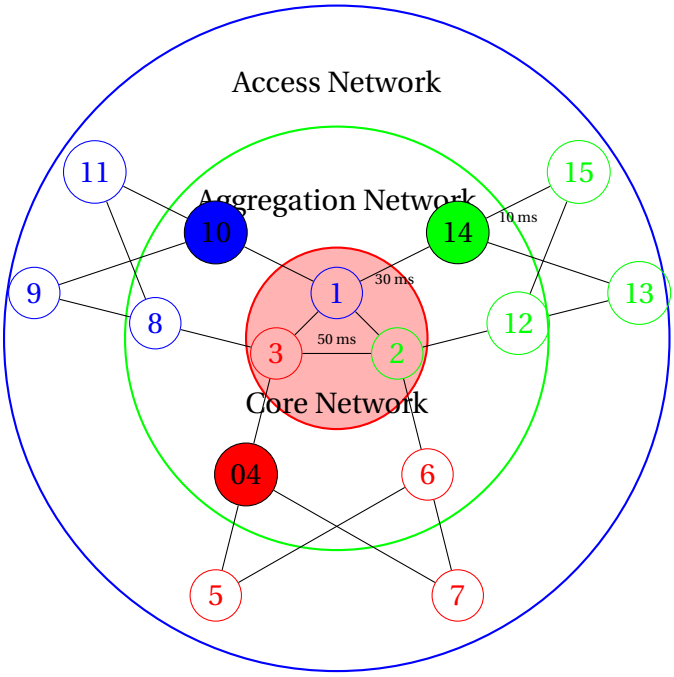


Figure 4.8 – Exemple de réseau Telco simplifié

4.3.8 Qualification sur une topologie WAN réaliste

Nous avons essayé de qualifier différentes versions de notre algorithme sur une topologie réelle. Nous avons choisi le WAN européen Geant tiré de la base de données zoo-topology. Nous avons testé le comportement de quatre versions de l'algorithme.

1. Une version basique sans mélange de la liste des nœuds, qui agrège au groupe courant le premier groupe rencontré qui vérifie la contrainte de latence et avec les coefficients α et β valant 0.5 et 0.9,

2. La même version dans laquelle est introduite un mélange aléatoire des nœuds répétés sur 100 itérations en choisissant ensuite le meilleur résultat,

3. Une version sans mélange aléatoire qui cherche à constituer les plus petits groupes possibles à chaque itération en choisissant les groupes les plus proches.

4. La même version avec mélange aléatoire sur 100 itérations et choix du meilleur résultat.

Le tableau 4.1 résume les résultats obtenus. Les coefficients α et β permettent de moduler le nombre de contrôleurs et d'améliorer ou de dégrader l'équilibre de charge. Ils sont fixés ici à 0.8 et 0.9. Les valeurs choisies ici tendent à réduire le nombre de groupes au détriment de la charge.

La conclusion importante de cette analyse est que le fait de privilégier les groupes les plus proches dans le déroulement de l'algorithme améliore les résultats, du point de

vue de l'équilibre de charge tout en augmentant le temps de calcul car il faut faire une comparaison entre tous les groupes. Bien sûr, le coût de l'amélioration de l'équilibre de charge est l'augmentation du nombre de groupes.

Variante	Résultats
Variante 1	Nb groupes: 2, Ecart max de charge: 32, variance de la charge: 512
Variante 2	Nb groupes: 2 Ecart max de charge: 30, variance de la charge: 450
Variante 3	Nb groupes: 6, Ecart max de charge: 10, variance de la charge: 11
Variante 4	Nb groupes: 7, Ecart max de charge: 4, variance de la charge: 4.5

Table 4.1 – Comparatif de différentes variantes sur le réseau Geant

Il est intéressant de visualiser le comportement de la variante 2 sur la topologie physique du réseau européen Geant en figure 4.9. Cette topologie est relativement hétérogène avec des nœuds isolés et éloignés. Dans ce scénario nous avons fixé la contrainte de latence à la moitié du diamètre du graphe calculé en utilisant la matrice des plus courts chemins. Intuitivement cela correspond à au moins deux groupes.

Compte tenu de la topologie hétérogène du réseau, il n'est pas surprenant qu'on trouve un fort déséquilibre de charge entre les deux groupes avec beaucoup plus de nœuds bleus que de nœuds rouges. Il faut remarquer aussi que si on réduit encore la latence d'un facteur 2.5 par exemple le nombre de contrôleurs augmente rapidement avec l'apparition de contrôleurs isolés.

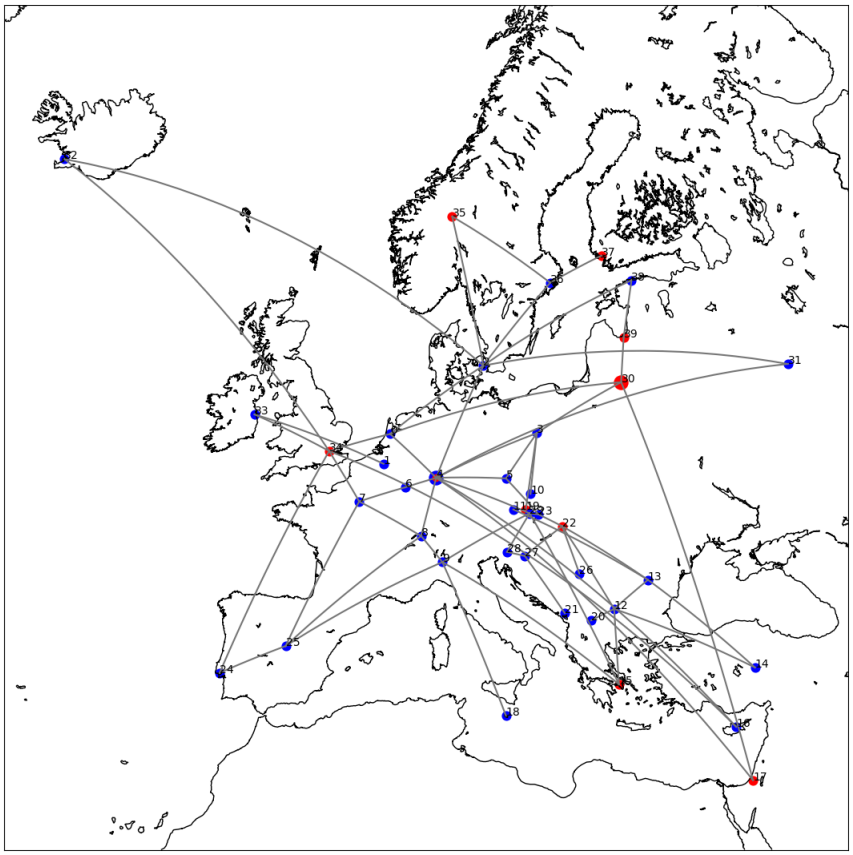


Figure 4.9 – Réseau Geant 2012

4.3.9 Caractéristiques des topologies WAN utilisées pour l'évaluation

Les données de topologies réseaux utilisées

Pour évaluer nos propositions d'algorithmes de placement, nous avons utilisé systématiquement la base de données zoo-topology[2].

Il s'agit d'une base de données de topologies réseaux de taille nationale ou pannationale, à l'échelle d'un continent voire mondiale. La base est constituée d'un ensemble de fichiers au format .gml, qui peuvent être manipulés à partir de logiciels de manipulations de graphes. Le format de fichier .gml permet d'attacher aux nœuds et aux liens du réseau des caractéristiques diverses comme la latence ou la charge de contrôle induite par le nœud contrôlé. Les fichiers de base fournissent pour une bonne partie d'entre eux les coordonnées géographiques des nœuds, mais pas de manière systématique. Nous avons complété la base de données de la manière suivante:

- Quand il manquait des coordonnées géographiques sur certains nœuds, nous leur avons attribué une coordonnée géographique moyenne complétée par la valeur d'une variable aléatoire uniforme afin d'éviter autant que possible des colocalisations.
- Nous avons attaché à chaque lien une latence estimée à partir des coordonnées géographiques des nœuds extrémités et en prenant en compte la vitesse de propagation de l'onde le long du lien.
- Quand le réseau est constitué de plusieurs composantes isolées, nous avons établi des liens artificiels entre des nœuds tirés aléatoirement des différentes composantes de manière à obtenir un réseau à une seule composante connexe. Cette légère modification simplifie l'évaluation des algorithmes.
- Bien que les topologies proposées soient orientées, nous les avons considérées systématiquement comme non orientées, c'est à dire que tous les liens sont bidirectionnels.

La base de données est constituée de 262 réseaux dont la taille varie de quelques nœuds à plusieurs centaines de nœuds et liens, comme pour le réseau Kdl.

Analyse des métriques des différentes topologies

Nous présentons ci-dessous quelques diagrammes sur les caractéristiques de la base de données des topologies. Elles sont calculées par rapport à 9 métriques différentes utilisées usuellement pour caractériser les graphes.

Le tableau 4.2 ci-dessous présente quelques caractéristiques statistiques de ces métriques calculées sur l'ensemble des réseaux.

Les critères calculés sont la moyenne, la variance, la médiane, la déviation par rapport à la médiane et les valeurs maximale et minimale de chaque métrique.

La densité d est le rapport du nombre de connexions possibles sur le nombre réel de connections dans le graphe. Elle s'exprime par la relation suivante dans laquelle n est le nombre de nœuds du graphe et m le nombre d'arcs.

$$d = \frac{\binom{n}{2}}{m} \tag{4.13}$$

[2] http://www.topology-zoo.org/

Le diamètre est le plus long chemin que l'on peut trouver entre deux nœuds du graphe, tandis que le rayon en est le plus court.

Le degré moyen est simplement le degré moyen de l'ensemble des nœuds du graphe, le degré d'un nœud étant le nombre d'arcs attachés au nœud, dans le cas d'un graphe non orienté.

La connectivité moyenne est établie en calculant la k-connectivité entre toutes les paires de nœuds divisées par le nombre de paires possibles comme dans l'expression de l'équation: 4.14.

$$\bar{k}(\mathrm{G}) = \sum_{(i,j),\ i<j} \frac{k_{\mathrm{G}}(i,j)}{\binom{n}{2}} \tag{4.14}$$

Le nombre de cliques est simplement le nombre de cliques contenus dans le graphe calculé grâce à un algorithme comme celui proposé par BRON et KERBOSCH [1973] que nous avons utilisé.

Le coefficient de groupage ("clustering") d'un nœud c_v est la proportion de formes en triangles qui existent dans le voisinage du nœud.

Le coefficient de groupage moyen est calculé avec l'expression suivante qui est le rapport du coefficient de groupage de chaque nœud sur le nombre de nœuds:

$$\mathrm{C} = \frac{1}{n} \sum_{v \in \mathrm{G}} c_v \tag{4.15}$$

Table 4.2 – Métriques des réseaux de la base zoo-topology

métrique	Moyenne	Dev	Médiane	Mad	Min	Max
Nb nœuds	36.33	28.96	27.50	17.05	4	197
Nb arcs	44.13	35.44	33	23.72	3	242
Densité	0.12	0.11	0.09	0.06	0.01	0.97
Diamètre	0.03	0.05	0.01	0.01	0.00	0.50
Radius	0.02	0.03	0.01	0.01	0.00	0.26
Degré moyen	2.01	0.61	2.00	0.00	1.00	7.00
Connectivité moyenne	1.36	0.53	1.22	0.31	1.0	7.58
Nombre de cliques	38.37	33.02	28.00	19.27	2.0	237
Coeff de groupage moyen	0.1	0.15	0.05	0.08	0.00	0.97

La figure 4.10, illustre le nombre de nœuds et de liens pour l'ensemble des réseaux de la base. On remarque qu'il y a une corrélation linéaire très forte entre le nombre de nœuds et de liens. Le calcul du coefficient de Pearson avec le logiciel R donne 0.976, ce qui confirme cette forte corrélation. Le réseau Kdl qui se compose de 794 nœuds et de 854 arcs a été extrait de la base car son nombre particulièrement élevé de nœuds justifie un traitement particulier, mais il confirme cette corrélation.

Dans cette logique, nous avons cherché plus systématiquement s'il existait des métriques prépondérantes dans une analyse visant à déterminer des classifications correspondants à des catégories de réseaux. Pour cela, nous avons effectué une analyse en composantes principales, illustrée dans le diagramme 4.11. On remarque de nouveau la forte corrélation entre le nombre de nœuds et le nombre d'arcs des réseaux (coefficient

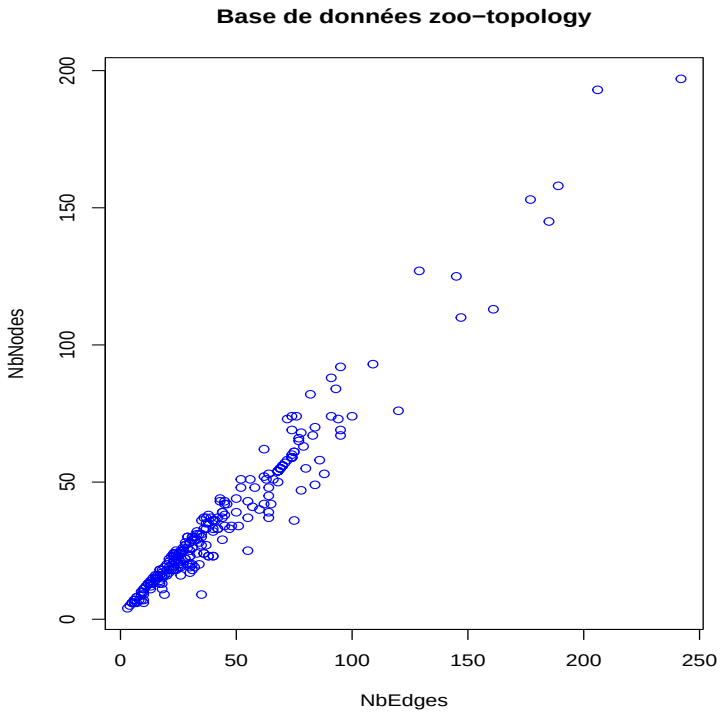


Figure 4.10 – Nombre de nœuds vs nombre d'arcs en %

de Pearson de 0.9), et une corrélation évidente entre le diamètre et le rayon du graphe (coefficient de Pearson de 0.99).

On remarque aussi une certaine corrélation entre des métriques qui peuvent refléter la connectivité, comme la connectivité moyenne, le degré moyen, le taux de clustering moyen, tandis que le nombre de cliques est très fortement corrélé au nombre de nœuds (coefficient de Pearson de 0.99).

Deux autres métriques sont relativement corrélées, il s'agit du degré des nœuds et de la connectivité moyenne, avec un coefficient de Pearson qui vaut 0.75.

Le taux moyen de groupage et la connectivité moyenne sont assez corrélés avec un coefficient de Pearson qui vaut 0.60.

Le taux moyen de groupage est corrélé avec le degré moyen avec un coefficient de Pearson de l'ordre de 0.69.

Du fait de nos travaux, nous avons regardé aussi la métrique de k-connectivité moyenne en fonction de la densité. Le coefficient de Pearson pour ces deux métriques est de l'ordre de 0.5, ce qui ne permet pas de distinguer une quelconque corrélation. Ce résultat est à prendre en compte car il semblerait indiquer que l'on ne peut pas utiliser la densité pour évaluer la connectivité et donc n'est pas en faveur de l'utilisation de l'algorithme de groupage Density-Based Spatial Clustering of Applications with noise (DBSCAN) [ESTER et collab., 1996] basé sur la densité.

4.3.10 Evaluation sur les différentes topologies de la base zoo-topologie

L'analyse des performances des deux algorithmes est effectuée sur toutes les topologies des réseaux inclus dans la base de données zoo-topologie détaillée en 4.3.9. Rappelons

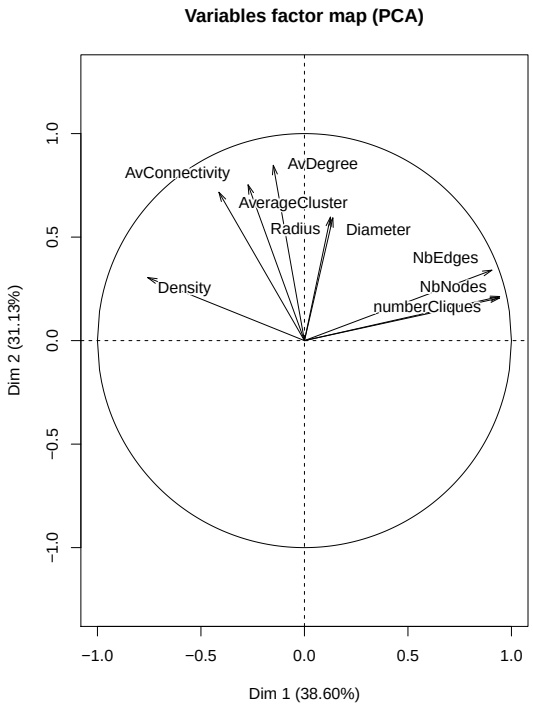


Figure 4.11 – Décomposition en composantes principales

la très grande variété de topologies représentées dans la figure 4.12 où chaque marque représente un réseau avec son diamètre et sa densité.

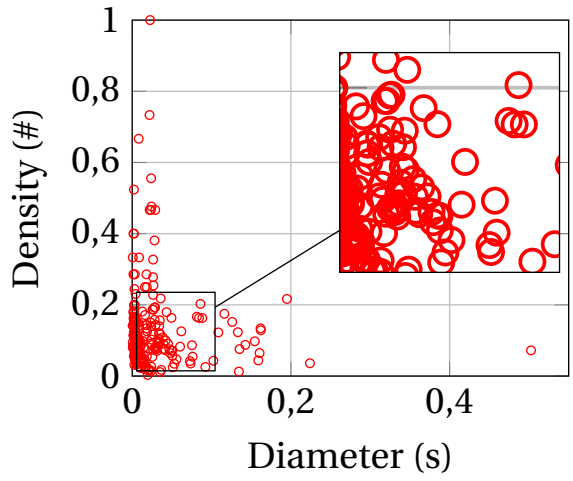


Figure 4.12 – Diversité des topologies réseaux sélectionnées

Nous nous concentrons maintenant dans cette section sur l'évaluation des performances à l'aide de simulations informatiques utilisant le langage `Python`. L'algorithme K-hiérarchique est comparé à l'algorithme K-mean-adapté et à la solution optimale dans le cas de petits réseaux.

Résultats obtenus avec K-mean adapté

Dans l'évaluation proposée sur cet ensemble de topologies, le critère de latence $l_{\max}$ est fixé à 12.5 ms. L'idée est de se placer dans les conditions nécessaires pour la mise en œuvre d'un service de re-routage rapide en garantissant que les contrôleurs seront placés de telle manière qu'une contrainte de latence de 50 ms pourra être globalement tenue. C'est le cas d'usage que nous avons décrit en 3.2.9, et c'est la norme dans les réseaux MPLS pour établir une route de secours.

Pour cela une valeur de 25 ms est fixée et attribuée aux traitements applicatifs répartis entre les SOFs (pour détecter une perte de lien) et le contrôleur pour effectuer les traitements applicatifs c'est à dire consulter la topologie et calculer une route de secours puis élaborer les commandes OF permettant d'établir la route de secours.

L'hypothèse de travail est celle d'un service de re-routage rapide pro-actif c'est-à-dire que les routes de secours ne sont pas pré-provisionnées ou établies préalablement mais sont calculées et établies à la demande. Enfin, les SOFs ne bénéficient pas de mécanismes particuliers de détection élaborés et doivent s'appuyer sur le contrôleur pour identifier un incident. C'est un scénario que pourrait potentiellement permettre un réseau SDN bénéficiant d'une vue centralisée du réseau.

Les 25ms sont le délai toléré aller et retour sur les plus courts chemins entre les SOFs et leur contrôleur. Ce qui correspond à une latence sur le chemin de 12.5ms. Il s'agit de la valeur seuil à partir de laquelle nous avons effectué notre analyse.

La figure 4.13 représente le nombre de contrôleurs (axe de gauche), et la variance sur la charge du groupe (axe de droite) calculée pour chaque réseau, en fonction du diamètre des réseaux, en appliquant l'algorithme K-mean adapté. L'algorithme est exécuté N fois en l'initialisant avec chaque nœud (m = N) du réseau et le résultat obtenu avec le plus petit nombre de groupes est sélectionné.

On peut remarquer que le nombre de contrôleurs tend à augmenter lorsque le diamètre du réseau augmente. La plage de valeurs de la variance sur la charge des groupes est cependant très large et peut être très différente selon les réseaux considérés. Enfin la variance diminue lorsque le diamètre du réseau est beaucoup plus grand que le critère de latence.

Ces résultats sont conformes à l'intuition. En effet, avec un diamètre de réseau large on peut s'attendre à avoir plus de contrôleurs et donc une diminution de l'écart type qui est réparti sur davantage de contrôleurs.

La figure 4.14 représente les résultats lorsque l'algorithme K-mean adapté est exécuté comme dans le diagramme précédent, mais cette fois la solution sélectionnée parmi les N itérations est celle qui présente la plus faible variation de charge entre les groupes. Nous constatons que l'amélioration n'est pas significative, car le nombre de groupes reste généralement le même avec une très faible amélioration de la variance sur la charge pour seulement quelques réseaux.

Résultats avec l'algorithme de groupage hiérarchique: K-hiérarchique

La figure 4.15 présente les résultats obtenus pour la version de base de l'algorithme K-hiérarchique, qui ne tient pas compte des contraintes de charge.

Pour cela, les ratios α et β sont désactivés lors de l'exécution de l'algorithme. On constate que les résultats obtenus sont très proches de ceux obtenus pour l'algorithme K-mean adapté en termes de nombre de contrôleurs. Une légère amélioration peut être notée pour la variance de la charge des groupes.

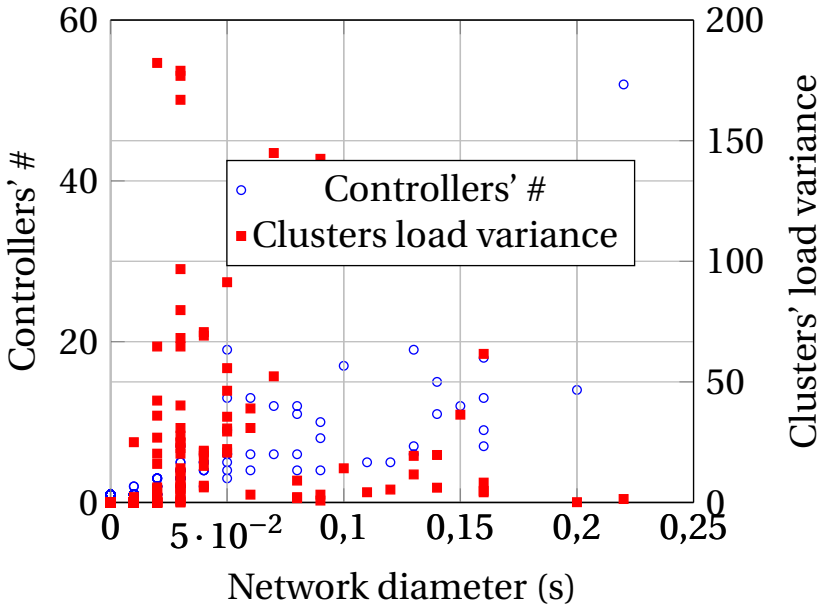


Figure 4.13 – Adapted K-mean: Nombre de contrôleurs et variance de la charge des groupes

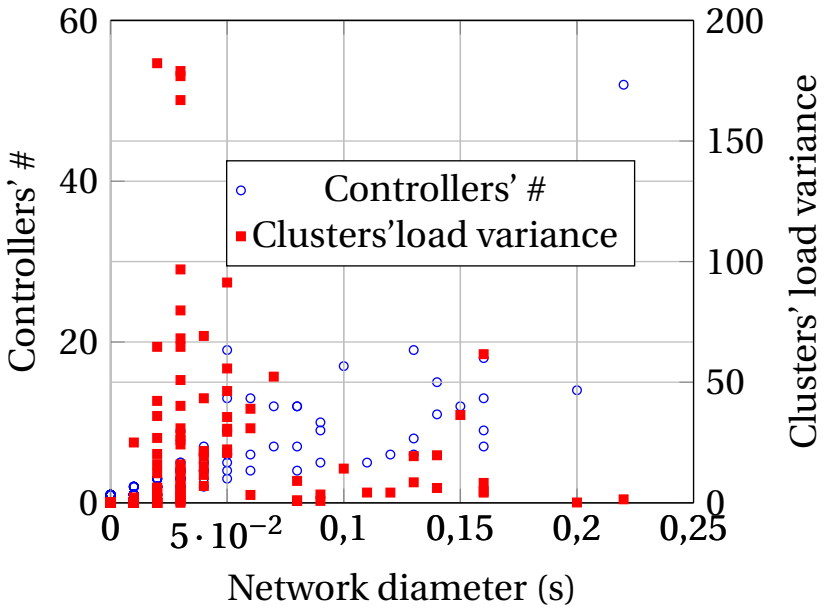


Figure 4.14 – Adapted K-Mean+: Nombre de contrôleurs et variance de la charge des groupes

Comme on peut le voir dans la Figure 4.16, les résultats peuvent être améliorés de manière significative en exécutant l'algorithme K-hiérarchique 100 fois avec une initialisation aléatoire de la liste des nœuds (c'est-à-dire en mélangeant chaque fois la liste des nœuds de manière aléatoire). Il suffit ensuite de choisir le meilleur résultat. L'exécution d'une initialisation aléatoire peut généralement permettre de trouver un meilleur optimum.

La figure 4.17 représente les résultats avec une version améliorée de l'algorithme (K-hiérarchique+) utilisant les paramètres de ratio α et β où $\alpha = 0.5$ et $\beta = 1$. Comme pour l'algorithme K moyen adapté, le nombre de contrôleurs de chaque réseau augmente avec le diamètre du réseau. De plus, le nombre de contrôleurs est supérieur à celui obtenu avec l'algorithme hiérarchique de base. A l'inverse, la variance de la charge des groupes est beaucoup plus faible et fortement aplatie car elle reste inférieure à 40 contrairement

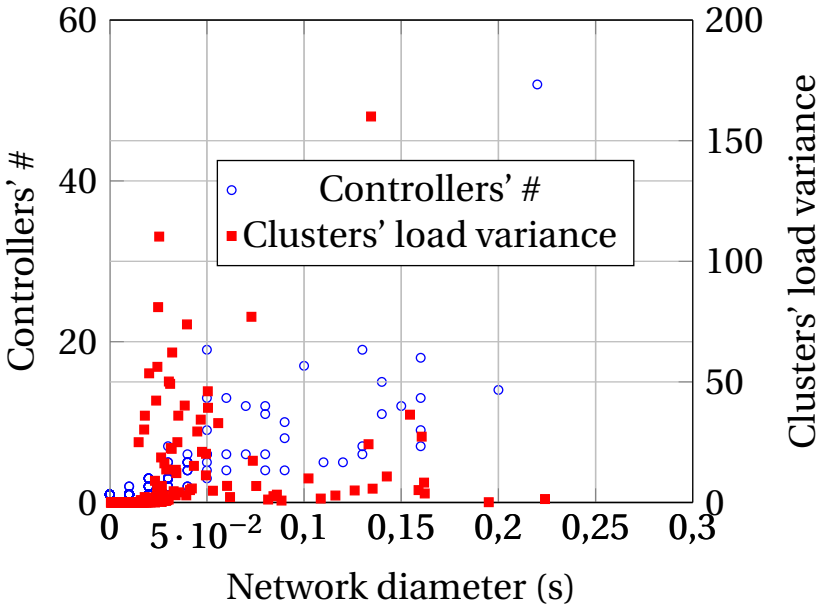


Figure 4.15 – K-Hierarchique (version basique): Nombre de contrôleurs et variance de la charge des groupes

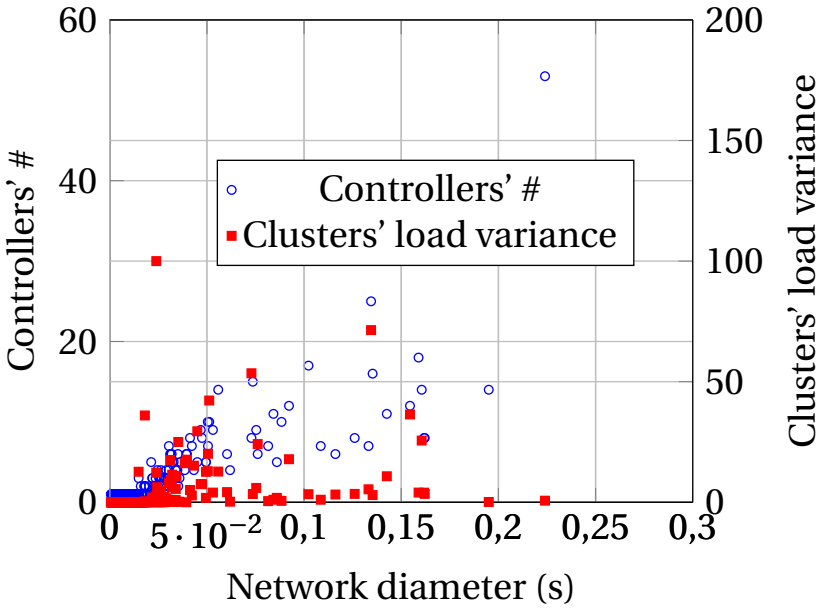


Figure 4.16 – K-hiérarchique (Initialisation aléatoire): Nombre de contrôleurs et variance de la charge des groupes

aux autres algorithmes.

Dans la suite, nous ne considérons que les versions de l'algorithme hiérarchique qui intègrent la répétition 100 fois de l'algorithme avec le mécanisme de mélange aléatoire à l'initialisation. Ce mécanisme améliore notablement les performances.

Afin d'améliorer notre compréhension de l'impact du réglage des paramètres de l'algorithme sur les résultats obtenus, nous analysons ci-après l'effet de plusieurs couples de α et β sur la performance.

Les figures 4.18 et 4.19 représentent le nombre moyen de contrôleurs et la variance moyenne de la charge des groupes pour tous les réseaux de la base de données zoo-topology pour des couples de α et β, avec des valeurs allant de 0.5 à 1.0 et un pas de

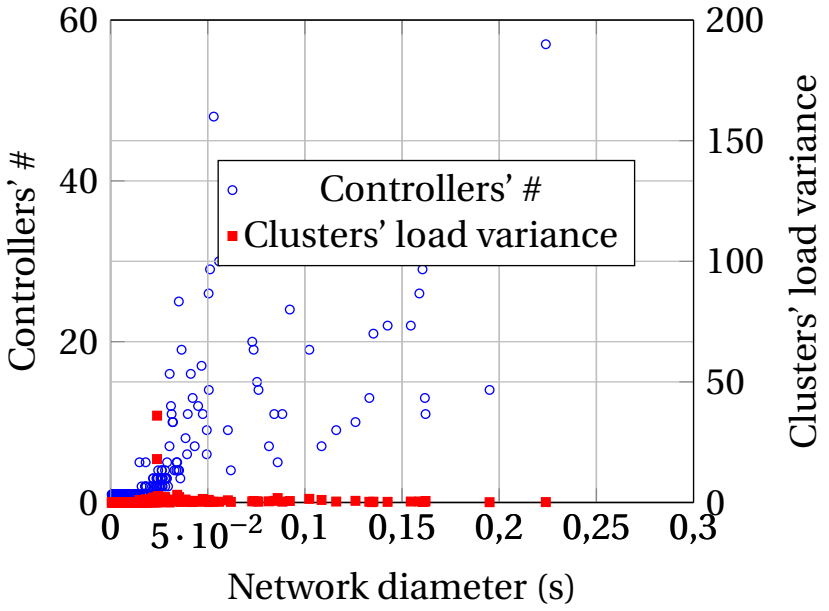


Figure 4.17 – Algorithme K-Hiérarchique+ avec $\alpha = 0.5$ et $\beta = 1$

0.05.

On peut constater que les deux paramètres agissent de manière contradictoire sur les indicateurs de performance clés considérés. Cependant, le paramètre α semble être le paramètre décisif.

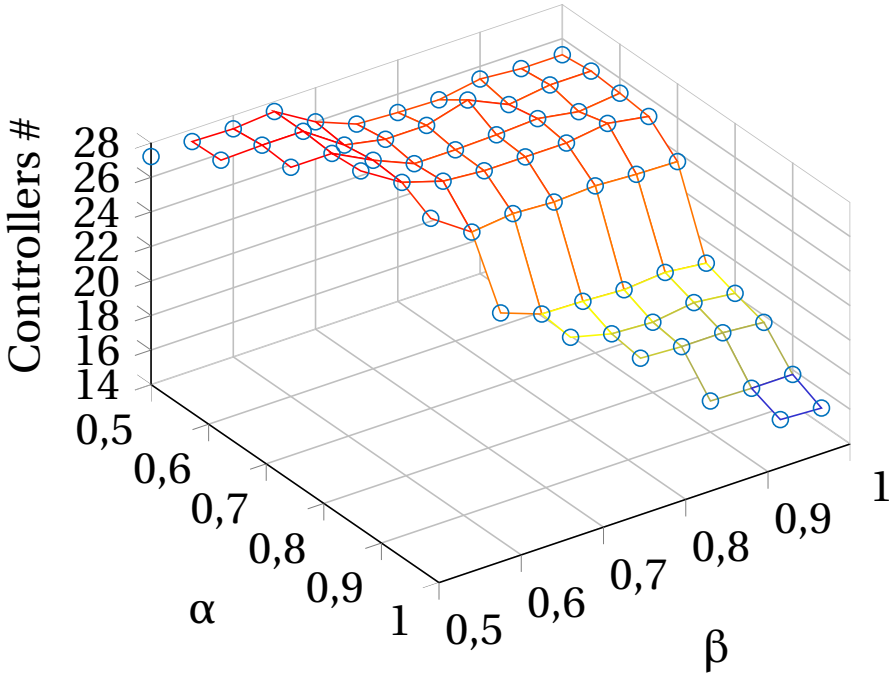


Figure 4.18 – Nombre moyen de contrôleurs en fonction α et β

Dans la figure 4.20, les points en rouge sont calculés sur tous les réseaux de la base de données zoo-topologie et correspondent aux valeurs de la variance moyenne de la charge moyenne des groupes et du nombre moyen de contrôleurs pour les différentes valeurs de α et β.

Ils sont principalement situés sur une courbe qui peut être facilement interpolée. Le

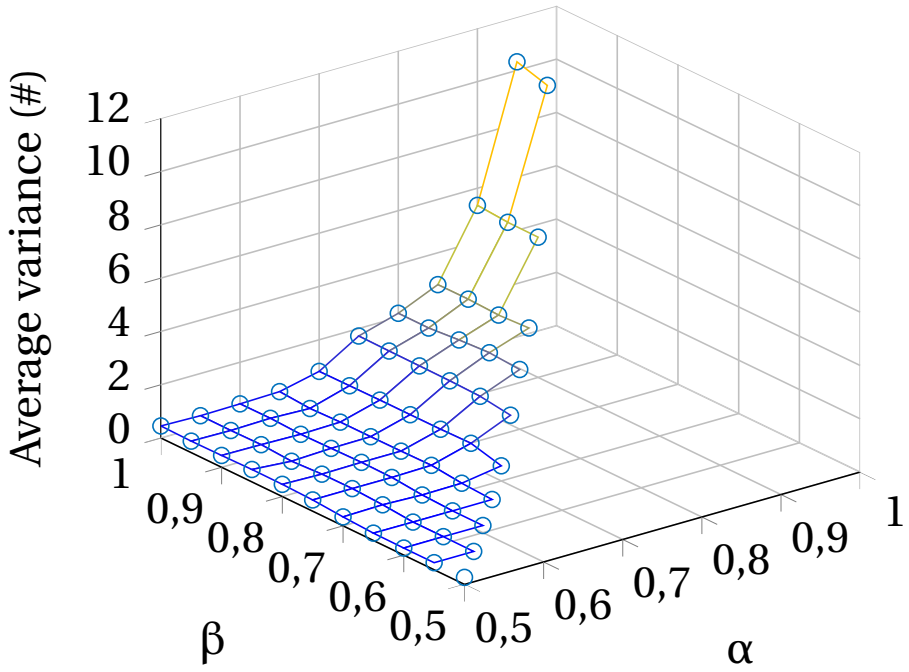


Figure 4.19 – Moyenne de la variance en fonction de α et β

compromis entre le nombre moyen de contrôleurs et la variance moyenne de la charge des groupes peut être décrit comme une fonction objective linéaire : J = $w_c * c + w_v * v$, où c est le nombre moyen de contrôleurs et v la variance de charge moyenne.

Dans ce modèle, w_c et w_v sont respectivement les poids donnés à c et v. Cette fonction correspond à une ligne droite de pente $-w_v/w_c$, qui croise en certains points la courbe d'interpolation.

Une méthode pour trouver le point optimal moyen correspondant à des poids choisis c et v est illustrée dans le diagramme.

En faisant glisser la ligne droite vers la gauche et en conservant sa pente jusqu'au dernier point rencontré, les paramètres α et β minimisant la fonction linéaire J pour les poids sélectionnés peuvent être dérivés en utilisant ensuite les deux graphiques 3D.

Enfin, la figure 4.21 présente les résultats obtenus en sélectionnant des valeurs de α = 0.8 et β = 0.9 qui correspondent au minimum de la fonction linéaire avec un poids équilibré pour le nombre moyen de contrôleurs et pour la variance moyenne (i.e. $w_v = 1$ et $w_c = 1$). Cela donne un nombre moyen de contrôleurs d'environ 18 et une variance moyenne de charge des groupes d'environ 2.0.

4.3.11 Conclusion

Nous avons montré qu'une heuristique basique de groupage pouvait aussi être spécialisée pour obtenir un outil de placement de contrôleurs dans une topologie réseau à la fois efficace et rapide.

Nous avons montré qu'elle pouvait être plus performante qu'une heuristique de type K-moyennes, couramment utilisée.

Nous avons vérifié la rapidité et l'efficacité de l'heuristique sur le réseau Kdl composé de 700 nœuds.

La structure et la logique d'un algorithme de groupage hiérarchique sont suffisamment simples pour pouvoir intégrer relativement simplement des boucles de contrôle

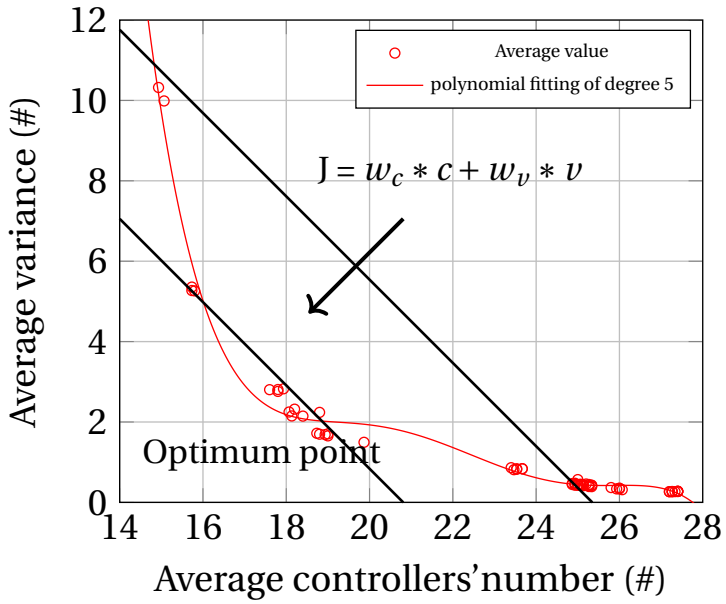


Figure 4.20 – Recherche du minimum de la fonction objectif linéaire J = $w_c * c + w_v * v$

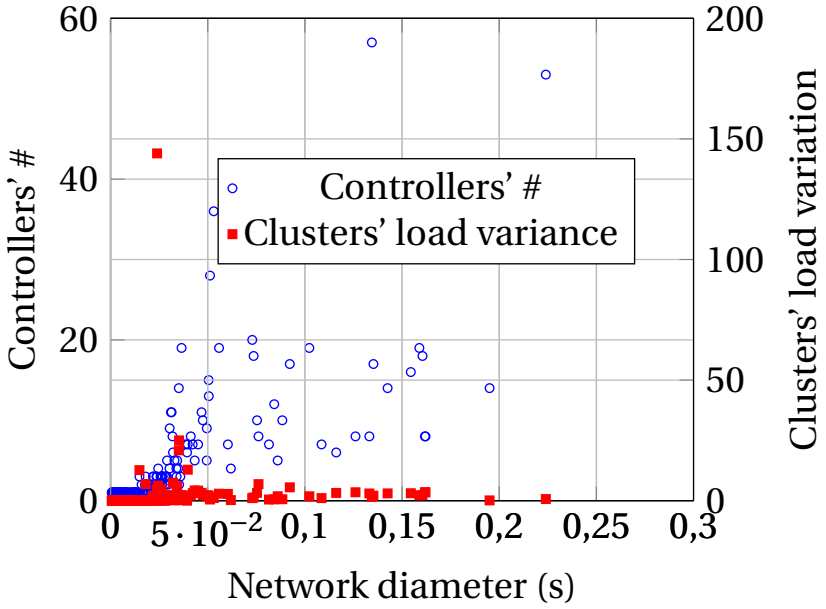


Figure 4.21 – Algorithme K-Hierarchique+ avec α = 0.8 et β = 0.9

pour intégrer des objectifs ou contraintes supplémentaires comme la charge des contrôleurs.

La comparaison avec un modèle ILP a montré que les résultats obtenus étaient proches de l'optimal pour de petites instances de réseaux.

4.4 Placement de contrôleurs avec un algorithme évolutionnaire

4.4.1 Choix de la structure de données

Nous présentons ici une réflexion préalable sur le choix de la structure de données adaptée pour le problème de placement des contrôleurs avec les AE.

Dans le cas d'un problème de placement de contrôleurs qui est un problème de groupage, le modèle doit décrire l'emplacement des contrôleurs et l'emplacement des SOFs qui lui sont attachés. Pour simplifier l'approche, on considère que les nœuds du réseau sont numérotés dans un ordre quelconque, comme dans le schéma 4.22 illustrant un réseau composé de 4 nœuds.

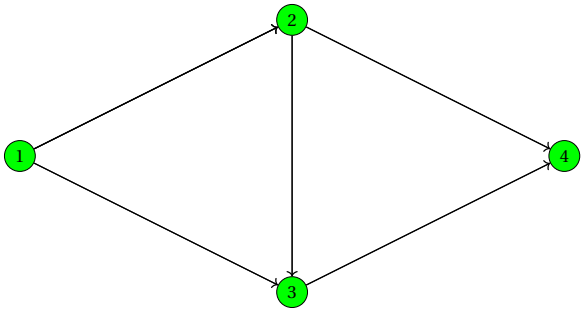


Figure 4.22 – Exemple de graphe avec nœuds numérotés

Si on adopte une représentation sous forme d'un mot binaire, une manière de procéder consiste à représenter une solution par un mot de longueur N, où N est le nombre de nœuds du graphe, et d'attacher à chaque nœud contrôleur la valeur 1. Mais cette représentation nécessite une structure de données supplémentaire ou imbriquée pour indiquer quels nœuds sont attachés à chaque contrôleur.

De plus, celle ci ne pourra pas être binaire sauf à définir une structure de données supplémentaire par contrôleur sous forme d'un mot binaire de longueur N de nouveau.

On adopte donc plutôt une structure sous la forme d'une liste de longueur N d'entiers, chaque entier désignant le contrôleur d'attachement comme dans la figure 4.23 avec deux contrôleurs numérotés 1 et 2.

En fait, si l'on remarque qu'un nœud contrôleur contrôle aussi sa fonction de routage local, un nœud contrôleur est simplement celui pour lequel le numéro du contrôleur correspond à son emplacement dans le réseau et dans ce cas une seule liste d'entiers est nécessaire.

Cependant, pour certains types de problèmes, il n'est pas forcément nécessaire de connaître à priori les routeurs attachés aux contrôleurs qui peuvent être retrouvés par un algorithme simple comme celui qui recherche les nœuds les plus proches de chaque contrôleur. Dans ce cas, une simple liste des nœuds contrôleurs peut suffire comme illustré dans la figure 4.24.

Une autre manière de procéder consiste à ne pas prendre en compte directement l'emplacement des contrôleurs et à considérer seulement les SOFs qui leur sont attachés, c'est-à-dire des groupes. On peut dans ce cas manipuler une liste d'ensembles d'entiers.

Le contrôleur correspondant à chaque groupe peut ensuite être calculé comme le centroïde de chaque groupe.

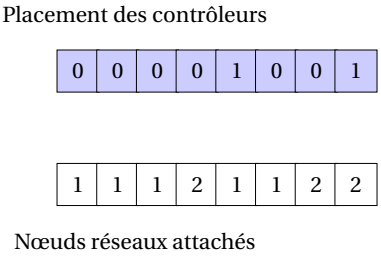


Figure 4.23 – Codage binaire ou par nombres entiers de l'emplacement des contrôleurs et des switches

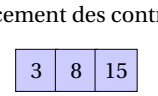


Figure 4.24 – Codage de l'emplacement des contrôleurs

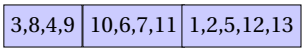


Figure 4.25 – Codage par groupes

4.4.2 Expérimentation d'un AE pour l'optimisation de la latence dans un réseau SDN

Dans cette contribution, nous avons cherché à aborder un problème de placement de contrôleurs relativement simple pour évaluer les performances que l'on peut attendre en mettant en œuvre un AE. Ce problème est suffisamment simple pour pouvoir être résolu aussi bien par une heuristique performante que par une méthode de programmation linéaire en nombres entiers.

L'intérêt de ce scénario est qu'il nous a permis de comparer les performances d'un AE avec un algorithme de programmation linéaire en nombres entiers qui produit une solution exacte. Nous avons évalué les résultats à partir de la base de données zoo-topology.

Le problème peut se formuler de la manière suivante: on dispose d'un nombre K de contrôleurs. On souhaite les placer dans le réseau de manière à minimiser la latence maximale entre les contrôleurs et les nœuds réseaux qui lui sont rattachés afin de garantir la meilleure réactivité possible des applications qui s'exécutent au niveau des contrôleurs.

G(V, E) est le graphe non orienté du réseau avec un ensemble de sommets V et un ensemble d'arcs E et leurs latences associées. Le nombre de contrôleurs est fixé à K.

Le but est de minimiser le diamètre maximum des groupes de nœuds autour des contrôleurs. On appelle diamètre la distance maximale entre le contrôleur et les nœuds qu'il contrôle au sens du plus court chemin. On appelle ici distance entre deux nœuds, le plus court chemin entre ces deux nœuds par rapport à la métrique de latence associée à chaque arc. Une donnée du problème est la matrice des plus courts chemins entre tous les nœuds du réseau qui est pré-calculée.

Définissons C comme l'ensemble des nœuds contrôleurs potentiels, où C est un sous-ensemble de V, on a $C \subseteq V$. Définissons également $c \in C$ comme un contrôleur potentiel et on note i_c un nœud de V attaché au contrôleur c. La distance (latence) entre un contrôleur c et un SOF i_c est désignée par $d(c, i_c)$.

Modèle linéaire en nombres entiers et Minizinc

Le problème d'optimisation se formalise simplement en écrivant:

$$\text{minimize} \max_{c \in \mathrm{C}, i_c \in \mathrm{V}} d(c, i_c) \tag{4.16}$$

Ce modèle se prête bien à un modélisation dans l'environnement Minizinc. Dans le modèle Minizinc, la représentation des solutions du problème et donc de la variable qu'on cherche à optimiser est un tableau des nœuds du réseau dans lequel à chaque nœud on fait correspondre son nœud contrôleur.

On introduit comme une contrainte le fait qu'un nœud contrôleur est piloté par lui même. Le nombre de contrôleurs fixé est lui même introduit comme une contrainte.

L'intérêt de cette approche est que la contrainte peut-être relâchée dans des problèmes d'optimisation où le nombre de contrôleurs n'est pas fixé et peut varier. Par exemple dans un scénario où on veut garantir un certain équilibre de charge entre les différents groupes.

Structure de l'algorithme évolutionnaire

Nous avons décidé de concevoir l'algorithme évolutionnaire dans une démarche *page blanche* afin de bien comprendre et d'explorer les possibilités de ce genre d'algorithme dans un scénario mono-objectif simple.

Modèle de données adopté

Plutôt que de se contraindre à un modèle construit avec un mot binaire et à une structure d'algorithme génétique standard, nous avons recherché un modèle de données pour les individus facilement utilisable dans l'environnement informatique Python que nous avons utilisé.

Nous avons choisi d'écrire un algorithme groupe-centrique, qui s'exécute sur des listes de groupes.

La structure d'un individu est une liste d'ensembles de nœuds, de type $[\{n_1, n_3, n_5\}, \ldots]$ chaque ensemble constituant un groupe de nœuds dont l'un parmi eux est le contrôleur du groupe. Le contrôleur du groupe n'est pas connu à priori mais peut-être facilement calculé à posteriori par une routine adhoc de type K-moyennes, K-centres ou K-médianes qui détermine le nœud le plus central en fonction de la manière dont on veut sélectionner le centroïde.

Présentation générale de l'algorithme

Paramètres d'entrée et de sortie: Les paramètres d'entrée de l'algorithme sont le graphe G(V, E) non orienté du réseau, le nombre de groupes égal au nombre de contrôleurs K, la taille de la population $popSize$, le nombre de générations $nbGen$, la pression de sélection p, les taux de mutations MUTPB et de cross-overs CXPB.

Les paramètres de sortie sont le meilleur individu (solution) trouvé, la latence maximale de la solution, qui correspond au diamètre du plus grand groupe.

On définit ici le diamètre du groupe comme la latence entre le nœud contrôleur et le SOF le plus éloigné du contrôleur au sens du plus court chemin.

Comme dans l'algorithme, on manipule des ensembles de nœuds, on doit à chaque fois connaître ou retrouver quel est le SOF qui est le contrôleur du groupe.

Pour cela on fait appel dans l'algorithme à deux routines optionnelles k-centres ou k-moyennes.

Algorithm 3: MAIN: Simple evolutionary algorithm

```
1:  INPUT: nbGen, popSize, netGraph, MUTPB, CXPB, k
2:  OUTPUT: BestIndividual, maxLatency

3:  MatriceSPF ⇐ SPF(netGraph)

4:  for index = 1 to popSize do
5:      population ⇐ randomClustering(k, netGraph)
6:  end for

7:  BestIndividual = ∅

8:  for index = 1 to nbGen do
9:      for individual ∈ population do
10:         Evaluate(individual)
11:     end for

12:     for index = 1 to popSize do
13:         parents ⇐ KTournamentSelection(population)
14:         child = cross-over(CXPB, parents)
15:         offSpring ⇐ child
16:     end for

17:     for individual ∈ offSpring do
18:         Mutate(MUTPB, individual)
19:     end for

20:     population ⇐ offSpring)
21:     BestIndividual ⇐ Update(BestIndividual, population)
22: end for
23: return BestIndividual,maxLatency
```

Initialisation: A l'initialisation de l'algorithme, on calcule en préalable la matrice de plus court chemin du graphe en entrée avec un algorithme de type Bellman-Ford.

Entre les étapes 3 et 5, une population initiale d'individus est constituée au hasard de taille $popSize$, via un mécanisme de tirage aléatoire avec remise dans l'ensemble des nœuds du réseau.

Chaque individu est alors constitué de k groupes qui couvrent le graphe du réseau.

La définition d'un individu est la suivante:

On note C_i un groupe d'indice i avec $i \in \{1..k\}$.

Un individu d de la population est décrit par un ensemble de k groupes qui couvrent l'ensemble des nœuds du graphe: $d = \cup_{i\in\{1..k\}}$ et on a:

$\cup_{i\in\{1..k\}} C_i = V$

ainsi que:

$\cap_{i\in\{1..k\}} C_i = \emptyset$

Evaluation: Entre les étapes 7 et 10, les individus sont évalués et on leur attache leur latence maximale. Il s'agit là de la latence maximale entre le contrôleur et les nœuds qui lui sont attachés parmi tous les groupes.

Elle est calculée avec l'une des deux méthodes possibles que nous avons intitulées : K-moyennes et K-centres. K-moyennes calcule le nœud contrôleur d'un groupe en cherchant la distance moyenne des nœuds du groupe au nœud contrôleur.

K-centres calcule le nœud contrôleur d'un groupe sur la base de la distance du nœud du groupe le plus éloigné du contrôleur, en la minimisant.

L'évaluation consiste à chercher le nœud contrôleur de chaque groupe et à attacher à l'individu la distance au sens du plus court chemin du nœud le plus éloigné de son contrôleur sur l'ensemble de tous les groupes.

Algorithm 4: K-MEAN: custom

```
1: Select all nodes as initial centroids
2: repeat
3:     Compute the average distance (spf) to all others nodes
4:     Keep the centroid which give the lowest average distance as controller
5: until End
```

Algorithm 5: K-CENTER: custom

```
1: Select all nodes as initial centroids
2: repeat
3:     Compute the max distance (spf) to all others nodes
4:     Keep the centroid which give the lowest max distance as controller
5: until End
```

Recombinaison: Le principe de l'opérateur de recombinaison que nous avons imaginé dans cette approche est de croiser K parents pour produire un enfant.

L'idée est d'adapter le mécanisme de croisement à notre cas particulier, où K est le nombre de groupes.

Avant cela, les K parents sont sélectionnés grâce à un opérateur de sélection par tournoi paramétré par la pression de sélection p. Le principe est de tirer aléatoirement avec remise p individus dans la population et de retenir le meilleur, puis de le faire autant de fois que l'on a besoin de parents pour renouveler la population.

Ensuite le croisement est effectué en tirant aléatoirement un groupe de chaque parent pour former un enfant composé des K groupes.

Comme les groupes de l'enfant peuvent avoir des nœuds communs ou qu'il peut manquer des nœuds à l'enfant qui a été produit, nous avons introduit dans l'opérateur de croisement une phase d'optimisation locale pour fabriquer un individu viable.

Cette phase d'optimisation permet de garantir que les groupes couvrent la totalité du graphe, et que les groupes constituent bien une partition des nœuds du graphe.

L'optimisation locale fonctionne de la manière suivante. On construit la liste des nœuds à ré-allouer, qui sont soit commun à plusieurs groupes, soit manquants pour couvrir la totalité du graphe.

On calcule le centroïde de chaque groupe et on attribue chaque nœud au groupe dont le centroïde est le plus proche.

Mutation: Un opérateur de mutation introduit ensuite de la variabilité en permutant de manière aléatoire certains nœuds des groupes, sur la base du taux de mutation en paramètre d'entrée de l'algorithme. Le taux de mutation détermine le nombre de nœuds tirés au hasard parmi tous les nœuds du graphe. Ces nœuds sont ensuite tirés au hasard avec une loi uniforme et un tirage sans remise et ils sont ensuite ré-alloués successivement, au hasard toujours, avec une loi uniforme parmi les groupes dans un nouveau groupe.

Renouvellement de la population: Dans l'algorithme, on effectue un remplacement générationnel, c'est-à-dire que l'ancienne population est entièrement remplacée par la nouvelle.

A chaque génération, on repère le meilleur individu et on le compare aux individus précédents. On conserve ainsi le meilleur au fil des générations.

Résultats obtenus et conclusion

Nous donnons ici quelques éléments sur la qualité des solutions obtenues que nous avons comparées aux solutions exactes obtenues grâce au modèle ILP sur l'ensemble des réseaux de la base de données zoo-topology décrite en 4.3.9. La taille de la population est de 50, le nombre d'itérations est de 100, le taux de mutation vaut 1% et la pression de sélection: 2.

- Pour 85% des réseaux, l'erreur relative par rapport à la solution exacte est inférieure à 10%.
- Pour 75% des réseaux, l'erreur relative est inférieure à 5%.
- Pour 65% des réseaux, l'algorithme évolutionnaire trouve la solution exacte. Pour 9 réseaux, l'écart relatif par rapport à la solution exacte est supérieur à 25%.

La qualité des solutions n'est pas corrélée au nombre de nœuds du réseau.

D'autre part, nous n'avons pas trouvé de métriques propres aux topologies de la base de données zoo-topology qui indiquent une corrélation avec la répartition de l'écart.

Dans le diagramme 4.26, les réseaux sont rangés sur l'axe des abscisses par taille en nombre de nœuds croissants, tandis qu'en ordonnée on mesure l'écart relatif de latence par rapport à la solution exacte obtenue avec le modèle ILP implémenté en Minizinc.

En terme de temps de calcul, l'approche ILP est capable de trouver la solution exacte en quelques secondes pour le réseau Kdl qui est composé de 700 nœuds. L'algorithme évolutionnaire a besoin de quelques dizaines de secondes.

Cependant notre implémentation est réalisée en Python et une implémentation en langage C permettrait probablement de gagner au moins un facteur 10. Dans ces conditions on a globalement un temps de calcul du même ordre de grandeur qu'avec la résolution du modèle ILP. Cependant, si l'on rajoute des contraintes au modèle ILP, comme par exemple, la recherche d'un équilibre de charge, dans ces conditions le temps de calcul explose. Alors qu'au contraire ajouter des contraintes à l'algorithme évolutionnaire n'augmente pas le temps de calcul.

Si l'on rajoute une simple contrainte d'équilibre de charge nous constatons que l'algorithme évolutionnaire est bien plus performant en terme de temps de calcul.

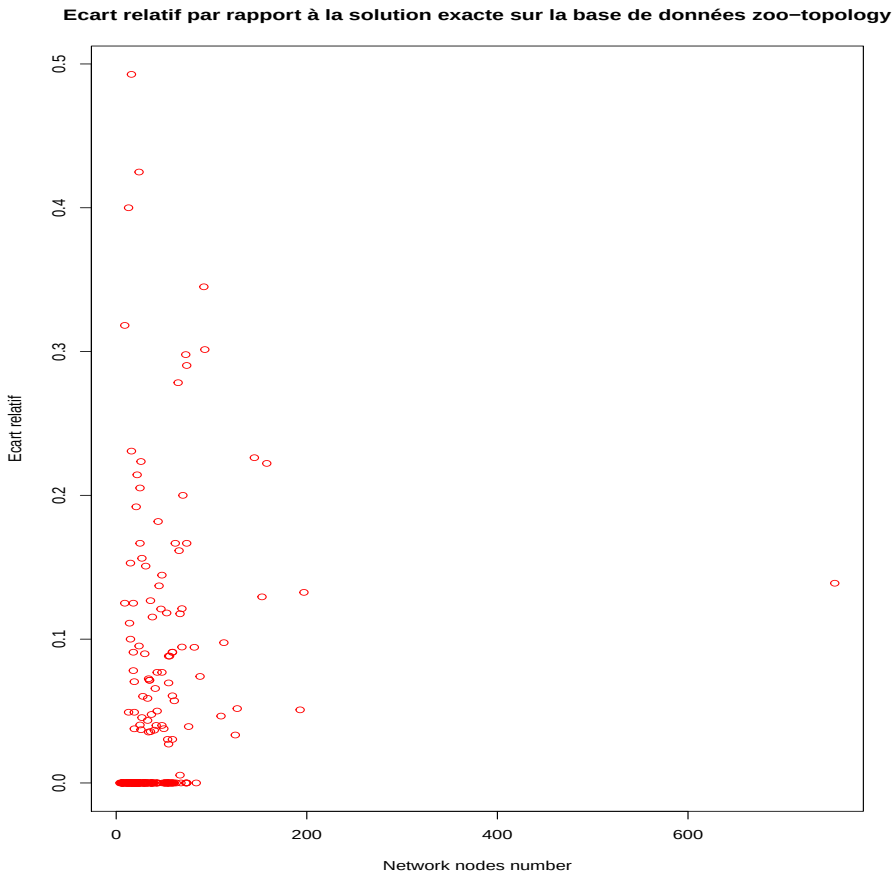


Figure 4.26 – Taille de la population: 50, nombre d'itérations: 100, Taux de mutation: 1%, Pression de sélection: 2

4.4.3 Placement fiable de contrôleurs avec un algorithme évolutionnaire

Introduction

Nos expérimentations sur un problème simple de placement de contrôleurs ont donné de bons résultats avec un algorithme évolutionnaire. Nous avons donc décidé d'explorer cette piste pour traiter des problèmes plus complexes notamment multi-objectifs. C'est ici aussi qu'a surgi l'idée d'utiliser une méta-heuristique pour traiter d'une manière plus générale les problèmes de placement de services réseaux. Nous nous sommes intéressés à la fiabilité des réseaux SDN et nous avons décidé de l'aborder du point de vue de la métrique de connectivité. Nous présentons de manière détaillée certains aspects de cette métrique dans le paragraphe suivant.

Fiabilité et métriques de connectivité dans les graphes

Rappelons tout d'abord quelques notions sur la mesure de la fiabilité des réseaux.

- On parle de fiabilité du terminal quand il s'agit de mesurer la fiabilité d'un nœud source qui communique avec un nœud terminal.
- On parle de k-fiabilité du réseau quand on s'intéresse à une communication de $k >$ 1 nœuds sources avec un nœud destination.

- On parle de fiabilité source à multiple quand il s'agit de considérer des communications d'un nœud source vers k nœuds destinations.
- On parle de fiabilité du réseau ou de fiabilité tous terminaux quand on considère des communications de tous les nœuds vers tous les nœuds du réseau.

D'autre part, il y a globalement deux manières de modéliser la fiabilité des réseaux. Ce sont d'une part les approches probabilistes basées sur l'attribution de probabilités de pertes sur les nœuds et les liens et d'autre part, les approches déterministes basées sur la connectivité qui cherchent à mesurer jusqu'à quel point un réseau est connecté.

L'idée est de compter le nombre de chemins disjoints composés de nœuds et de liens qui raccordent les nœuds du graphe.

Plus le graphe est connecté plus il est susceptible de résister aux pannes dans la mesure où des chemins alternatifs peuvent prendre le relais de chemins défectueux.

Nous avons choisi de nous intéresser dans notre problématique de placement de contrôleurs à une approche déterministe et à la mesure de la connectivité dans les graphes.

Des éléments théoriques et des pré-requis sur cette notion sont présentés en annexe.

Nous soulignons dans la suite du texte quelques points importants.

Passage à un graphe non-orienté Nos modèles de réseaux sont généralement des graphes non orientés.

Pour implémenter le modèle linéaire en nombres entiers permettant de calculer la connectivité pour un graphe non-orienté, la solution habituellement adoptée consiste à transformer le graphe non-orienté en un graphe orienté.

Il faut pour cela transformer le graphe en un graphe orienté en représentant un arc non-orienté par deux arcs orientés.

Nous avons préféré adopter une stratégie différente pour simplifier la mise en œuvre de solveurs en nombres entiers sur les topologies que nous avons manipulées.

L'idée consiste à attribuer une capacité qui peut autoriser un flot valant: -1 ou 0 ou 1 à tous les arcs non-orientés, le signe permettant de caractériser le sens du flot.

Le modèle linéaire permet alors de trouver la coupe de capacité minimum et donc la k-arc-connectivité pour un graphe non orienté.

Il suffit pour le vérifier de remarquer qu'avec ces valeurs de flots, un arc non-orienté est représenté par deux arcs orientés.

Les couples de valeurs de flots sur chaque paire d'arcs orientés représentant un arc non orienté peuvent être $(1,0),(0,1)$ ou $(1,1),(0,0)$. Cela correspond à une valeur de flot sur l'arc non orienté qui pourra valoir $c = 1$, $c = 0$, $c = -1$.

En effet, comme les deux arcs orientés sont de sens contraire, une valeur de $c = 1$ sur l'arc non incident au nœud et de $c = 0$ sur l'autre correspond à une valeur de flot de $c = -1$ sur l'arc non orienté, et une valeur de $c = 1$ sur l'arc incident et de $c = 0$ correspond à une valeur globale de flot de $c = 1$ sur l'arc non orienté.

Le schéma 4.27 permet d'illustrer le raisonnement.

Différence entre k-arcs-connectivité et k-nœuds-connectivité Nous avons eu dans notre travail quelques difficultés à faire la distinction entre arcs-connectivité et nœuds-connectivité dans un graphe.

En pratique, on peut simplement remarquer que si deux nœuds sont adjacents, la arcs-connectivité doit prendre en compte l'arc reliant les deux nœuds tandis que la nœuds-connectivité ne peut pas être calculée, puisqu'on ne peut pas enlever de nœuds pour les déconnecter.

Figure 4.27 – Illustration graphique

Le schéma 4.28 permet de voir sur un exemple simple qu'il n'est pas si évident de faire un lien entre arcs-connectivité et nœuds-connectivité entre deux nœuds u et v.

On remarque dans ce schéma qu'il faut enlever deux arcs pour déconnecter les nœuds u et v et que cela correspond à deux chemins indépendants qui ne partagent pas d'arcs communs) entre u et v. Tandis qu'en enlevant le nœud 3 on déconnecte u et v, mais il n'y a qu'un seul chemin indépendant entre u et v (qui ne partagent pas de nœuds communs).

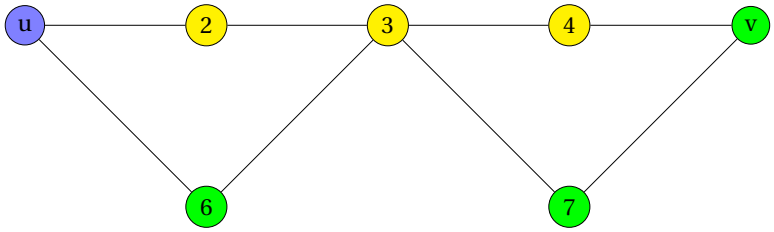


Figure 4.28 – Exemple simple

En résumé, il ne semble pas y avoir de relation évidente ou triviale entre arcs-connectivité et nœuds-connectivité.

Le calcul de la nœuds-connectivité entre deux nœuds non adjacents d'un graphe est cependant basé lui aussi sur le théorème du flot maximum. Pour le réaliser, on transforme le graphe orienté en remplaçant chaque nœud par deux nœuds joints par un arc orienté et on applique un algorithme similaire à celui utilisé pour traiter le cas de la arc-connectivité.

Insuffisances de la k-connectivité d'un graphe La k-connectivité d'une paire de nœuds (u, v) d'un graphe G se note $k_G(u, v)$ et celle d'un graphe G se note k(G) et peut s'exprimer comme dans l'équation 4.17

$$k(\mathrm{G}) = \min_{\forall (u,v) \in \mathrm{V}} k_G(u, v) \tag{4.17}$$

Il est assez facile de voir les limites de la k-connectivité d'un graphe comme outil pour évaluer la fiabilité d'un réseau, en effet cette métrique reflète d'abord la pire des situations. On voit par exemple sur le schéma 4.29 que la k-arcs connectivité vaut 1, puisque le nœud isolé est raccordé par un seul lien au reste du réseau et donc qu'en enlevant ce lien on déconnecte le graphe. Dans ce cas la k-arcs-connectivité du graphe vaut simplement 1 et ne reflète pas bien la connectivité globale du réseau et en particulier ne met pas en évidence la clique constituée par les nœuds 2, 3, 4, 5.

Intérêt de la connectivité moyenne d'un graphe pour estimer la fiabilité Il existe un certain nombre de métriques plus pertinentes que la k-connectivité. L'une d'elle est la connectivité moyenne décrite par [BEINEKE et collab., 2002b].

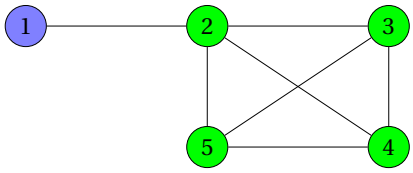


Figure 4.29 – Insuffisance de la k-connectivité

En utilisant une énumération de toutes les paires de nœuds du graphe, on peut exprimer la k-connectivité moyenne $\bar{k}(\mathrm{G})$ du graphe G d'ordre m comme la connectivité moyenne calculée sur toutes les paires de nœuds du graphe:

$$\bar{k}(\mathrm{G}) = \sum_{(i,j)\in \mathrm{V}^2} \frac{k_{\mathrm{G}}(i,j)}{\binom{n}{2}} \tag{4.18}$$

L'expression au numérateur de l'équation 4.18 est appelée connectivité totale du graphe. Comparativement à la k-connectivité, la k-connectivité moyenne exprime l'espérance d'une variable aléatoire qui est le nombre d'arcs enlevés au hasard nécessaires pour déconnecter le graphe.

Les deux graphes de la figure 4.30 permettent d'illustrer l'intérêt de la métrique de connectivité moyenne.

Le graphe de gauche G1 avec les nœuds en vert est un arbre et sa k-connectivité ainsi que sa connectivité moyenne valent 1. Les deux métriques permettent ici d'illustrer la connectivité du graphe.

Le graphe de droite G2 avec les nœuds en bleu a une k-connectivité qui vaut 1 du fait d'une branche isolée, tandis que sa k-connectivité moyenne vaut 2.2.

En conclusion, la connectivité moyenne reflète bien mieux la connectivité du graphe.

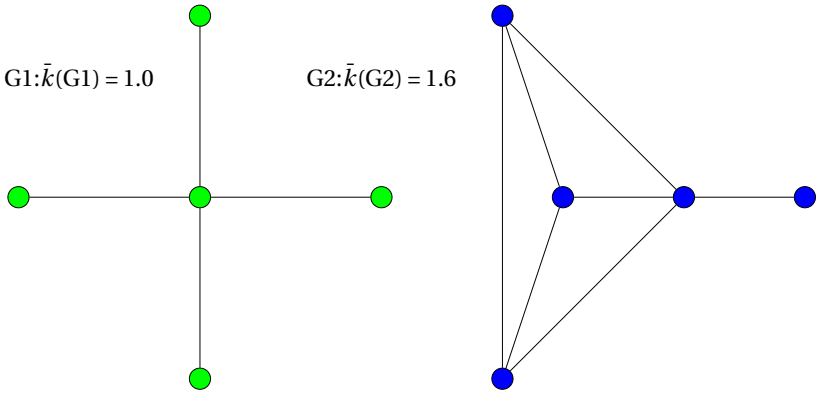


Figure 4.30 – Supériorité de la connectivité moyenne par rapport à la k-connectivité (exemple 1)

On peut cependant supposer que la connectivité moyenne est meilleure du fait du nombre d'arcs plus importants dans le graphe G2.

Ce n'est pas le cas comme le montre le schéma 4.31 ou $\bar{k}(\mathrm{G1}) = 1.3$ tandis que $\bar{k}(\mathrm{G2}) = 1.6$ avec le même nombre d'arcs dans les deux graphes.

Quelques règles triviales sont rappelées ci-dessous pour le calcul de la connectivité moyenne.

- $\bar{k}(\mathrm{G}) = 0$ si et seulement si G n'a pas d'arcs,
- Si G est un arbre avec au moins deux nœuds, $\bar{k}(\mathrm{G}) = 1$,

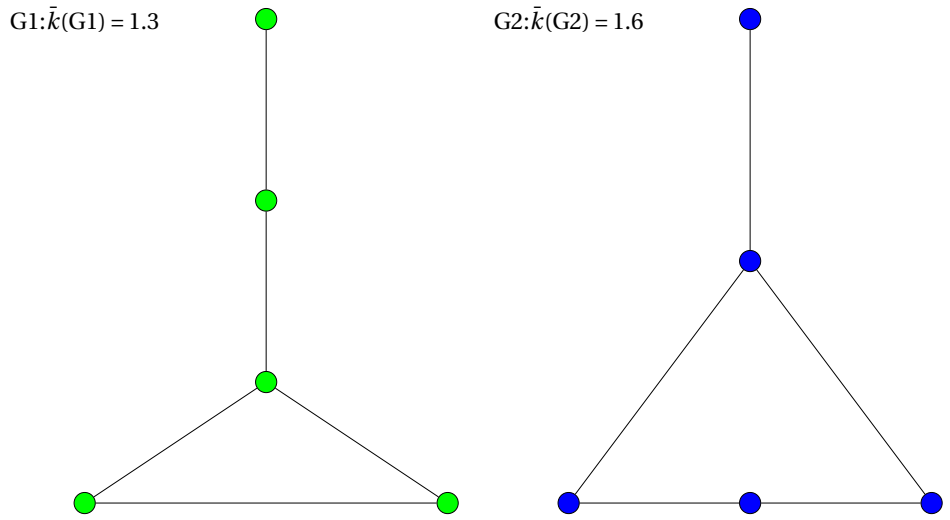


Figure 4.31 – Supériorité de la connectivité moyenne par rapport à la k-connectivité (Exemple 2)

- Si G est d'ordre p, $\bar{k}(G) \leq p-1$,
- Si G est d'ordre p et complet, $\bar{k}(G) = p-1$

Les deux exemples de la figure 4.32 permettent de montrer l'intérêt de la métrique de connectivité moyenne pour diviser un graphe en sous-graphes.

Dans les deux graphes, la moyenne de la connectivité moyenne des sous graphes G1 et G2 est supérieure à la connectivité moyenne du graphe G.

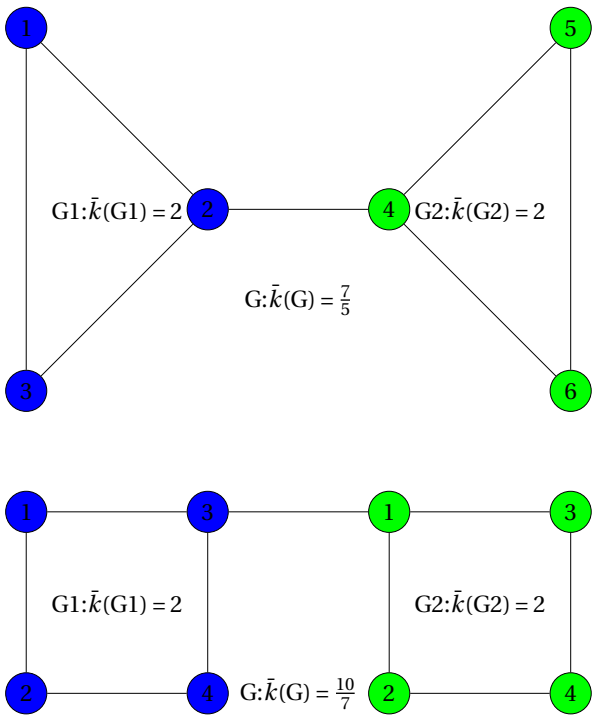


Figure 4.32 – Supériorité de la connectivité moyenne par rapport à la k-connectivité (Exemple 3)

Justification de la connectivité moyenne par passage à une approche probabiliste
Le passage à une approche probabiliste n'a rien d'évident, cependant on peut vérifier

l'intérêt d'utiliser la connectivité moyenne dans un réseau SDN en montrant le gain obtenu via une approche probabiliste en utilisant un modèle simple et des exemples. Le raisonnement s'inspire d'un modèle proposé par [BAR-ILAN et PELEG, 1991a].

On considère que les probabilités de pertes sur les nœuds ou les liens sont homogènes et indépendantes. On pose P_e la probabilité de perte sur un lien et P_n la probabilité de perte sur un nœud.

On suppose qu'un nœud SOF est connecté à son contrôleur via le chemin le plus court "shortest path first" qui le relie au contrôleur.

On fait l'hypothèse qu'un nœud n'est plus opérationnel lorsqu'il n'est plus connecté à son contrôleur. On note e_i les arcs sur le chemin qui relie le SOF au contrôleur et n_j les nœuds sur ce même chemin et p_{e_i}, p_{n_j}, les probabilités de pannes associées à l'arc e_i et au nœud n_i.

La probabilité de perte de communication du SOF par rapport à son contrôleur s'exprime de la manière suivante dans l'équation 4.19, où l'on parcourt le plus court chemin entre le contrôleur et le SOF.

$$P_{sof} = 1 - \prod_{i,j} (1 - p_{e_i}) \cdot (1 - p_{n_j}) \tag{4.19}$$

On peut évaluer le risque de dysfonctionnement du réseau SDN en prenant comme vraisemblance la moyenne de la probabilité de perte de communication entre les SOFs et leur contrôleur, calculée sur tous les SOFs raccordés au contrôleur par un plus court chemin, et chercher le contrôleur dans un réseau ou dans un sous-graphe qui minimise cette vraisemblance. C'est un problème d'optimisation.

On modélise le réseau comme un graphe G(V,E) avec |V| = N, G est non orienté et connexe. On pose n_c le nœud contrôleur, $n_{i:1..N-1}$ les SOFs contrôlés par le contrôleur n_c.

Les n_i sont connectés à n_c par des plus courts chemins P_i.

On pose comme vraisemblance de déconnexion, $\mathbb{E}(n_c)$, la probabilité moyenne de déconnexion des SOFs n_i connectés à n_c.

Les probabilités de pannes des nœuds et liens du réseau sont indépendantes et on pose pour simplifier:

Pour tous les nœuds n_i:

$p_{n_i} = p_n$, $\forall i$,

et pour tous les arcs e_i:

$p_{e_i} = p_e$, $\forall j$.

La probabilité de déconnexion d'un chemin P_i est donnée dans l'équation 4.19,

et la vraisemblance vaut:

$\mathbb{E}(n_c) = \frac{1}{N} \cdot \sum_i P_i$

Le problème d'optimisation consiste à trouver le nœud contrôleur n_c qui minimise $\mathbb{E}$, ce qui s'exprime par l'expression suivante:

$$n_c = arg_{n_i} \min \mathbb{E}(n_i) \tag{4.20}$$

qui consiste à trouver le nœud n_i contrôleur pour lequel $\mathbb{E}(n_i)$ est minimum.

On peut par exemple effectuer cette recherche de manière systématique grâce à un petit programme écrit en python sur le réseau illustré en 4.33 qui est composé de deux cliques de 4 nœuds reliées entre elles par un seul lien.

L'hypothèse est que deux contrôleurs (un par clique) permettront d'obtenir une meilleure fiabilité qu'un seul pour l'ensemble du réseau.

Le programme recherche dans ce réseau le nœud contrôleur qui minimise une vraisemblance exprimée comme la probabilité moyenne de pertes de communication pour tous les autres nœuds vus comme des SOFs auxquels il est connecté.

Il utilise pour cela une sous-routine qui calcule le plus court chemin entre les paires de nœuds et donc entre tous les nœuds contrôleurs potentiels et tous les SOF potentiels.

Dans le cas de ce réseau simple, si on prend par exemple $p_e = 0.02$ et $p_n = 0.01$, on trouve un nœud contrôleur pour lequel la probabilité moyenne de perte de communication moyenne est minimale et vaut: 0.0517.

Si on effectue le même calcul et la même recherche sur l'une des cliques de connectivité moyenne plus élevée en l'isolant comme un domaine piloté par un contrôleur on trouve 0.0395. Comme les deux cliques sont identiques, on obtient en moyenne la même valeur sur l'ensemble du réseau.

On voit donc qu'on a intérêt à subdiviser le réseau en deux composantes connexes pilotées chacune par un contrôleur et qu'en faisant cela on améliorera ainsi la fiabilité globale du réseau SDN, telle que nous l'avons définie dans ce modèle.

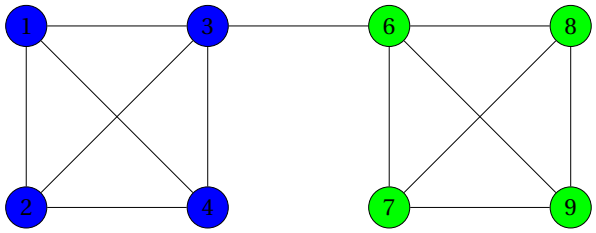


Figure 4.33 – Approche probabiliste

Nous avons effectué une vérification du même type sur un réseau réel, le réseau Geant2012 de la base de données zoo-topology illustré en 4.34.

L' algorithme évolutionnaire que nous présentons dans la suite trouve la solution en rouge et bleu et propose une solution avec deux clusters équilibrés dans le front de Pareto.

Les meilleurs nœuds contrôleurs au sens du problème d'optimisation que nous venons de décrire, sont indiqués par un gros carré bleu pour le cluster bleu, un gros carré rouge pour le cluster rouge. Le gros carré noir indique le contrôleur pour le réseau global.

Le même calcul que dans le cas d'école précédent donne, dans le cas du contrôleur global une probabilité moyenne de perte de communication de 0.0769, tandis qu'on obtient 0.0663 pour le contrôleur du cluster rouge et 0.0723 pour le contrôleur du cluster bleu.

On a donc une légère amélioration de la fiabilité du réseau SDN en partitionnant le réseau suivant les deux clusters.

Il faut noter ici que les clusters sont géographiquement imbriqués du fait que l'on ne cherche pas dans l'algorithme à regrouper autour d'un contrôleur suivant un critère de latence. Cet objectif supplémentaire pourrait être intégré dans l'algorithme. C'est d'ailleurs un des arguments qui en justifie l'intérêt.

Arc-connectivité moyenne & nœud-connectivité moyenne Pour implémenter notre AEMO sur des réseaux réels il serait pertinent de prendre en compte aussi bien la arc-connectivité moyenne que la nœud-connectivité moyenne, puisque autant la défection d'un lien que d'un nœud peut provoquer des pertes.

Comme il n'y a pas de lien direct entre les deux métriques, une des solutions pourrait être de rajouter une fonction objective à notre AEMO pour prendre en compte la nœud-connectivité moyenne.

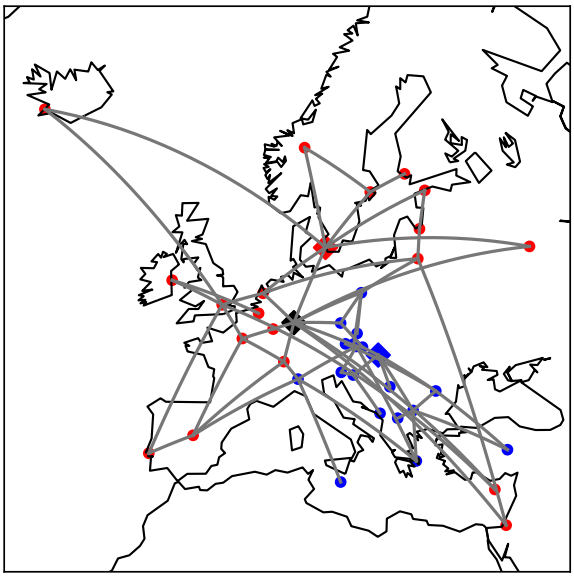


Figure 4.34 – Réseau Geant2012

Nous suggérons une approche différente qui consiste à réaliser une transformation du graphe du réseau de la manière suivante. Tous les arcs orientés avec des attributs (pertes, bande passante) sont transformés en un nœud avec les mêmes ressources. Les connexions entre les nœuds sont sans pertes, sans latence et avec une bande passante infinie.

Dans ces conditions il n'est plus nécessaire sur ce nouveau graphe de raisonner sur la arc-connectivité mais uniquement sur la nœud-connectivité, qui regroupe une métrique qui inclue l' arc-connectivité et la nœud-connectivité du graphe initial.

Description du système

Le système que nous considérons ici est l'architecture d'un réseau Telco dans lequel les éléments de réseau sont pilotés par quelques contrôleurs SDN distribués. Ces derniers sont responsables du plan de contrôle, qui pilote l'exécution des services réseau, tandis que les nœuds du réseau sont responsables des fonctions du plan de transfert.

De la manière la plus générale possible, nous considérons dans ce qui suit, une architecture réseau où chaque contrôleur peut être situé sur n'importe quel nœud du réseau. Par ailleurs, notre objectif lors du déploiement des services réseau est de choisir commodément le nombre de contrôleurs tout en les localisant de manière optimale dans le réseau de manière à respecter les contraintes des services réseau.

Comme nous l'avons indiqué, beaucoup de métriques peuvent être considérées, ici nous nous intéressons à:

- **Une arc-connectivité moyenne élevée:** La arc-connectivité moyenne donne une meilleure image de la fiabilité du réseau, puisqu'elle fournit une mesure de la "quantité" globale de connectivité d'un graphe selon BEINEKE et collab. [2002b].

Ainsi, en maximisant la connectivité moyenne des groupes, nous espérons maximiser le nombre de chemins disjoints entre les nœuds d'un groupe et leur contrôleur [BAR-ILAN et PELEG, 1991a] et donc améliorer la fiabilité globale du réseau SDN.

- **Un bon équilibre de charge entre les contrôleurs:** En minimisant le déséquilibre du trafic de contrôle entre les groupes, nous espérons réduire la latence entre les contrôleurs et les switches à l'intérieur d'un groupe. Cela permet également de réduire les risques de surcharge des contrôleurs.

- **Réduire le nombre de contrôleurs:** En minimisant le nombre de contrôleurs donc le nombre de groupes, nous cherchons à réduire les coûts d'exploitation.

Description du modèle

Notre modèle vise à fournir le cadre théorique de résolution d'une configuration de groupes de nœuds du réseau qui maximise la somme de la arc-connectivité moyenne des groupes tout en minimisant le déséquilibre entre la charge des groupes. Le modèle est basé sur la programmation linéaire, et utilise le théorème du flot maximum ou encore de la coupe minimum comme décrit par [SAKAROVITCH, 1984].

Soit G(V, E) un graphe non orienté, où V est l'ensemble des nœuds du réseau avec |V| = m et E l'ensemble des arcs avec |E| = n.

Soit $c(u_i) \in \{-1, 0, 1\}$ une fonction de capacité des arcs (à l'exception de l'arc de retour u_r pour lequel $c(u_r) = \infty$).

Le programme linéaire 4.21 exprime le flot maximum entre un nœud source s et un nœud destination d et donc la arc-connectivité entre les deux nœuds dans le graphe G.

$$\begin{aligned} \underset{f}{\text{maximize}} \quad & f(u_r) \\ \text{subject to} \quad & \text{A}.f = 0 \end{aligned} \tag{4.21}$$

Où A est la matrice d'incidence de G, u_r est un arc de retour fictif entre s et d qui est utilisé pour calculer le flot maximum.

De plus, f est un vecteur $\mathbb{R}^n$ qui représente le flot avec $f(u_i) \leq c(u_i)$ pour tous les u_i.

La matrice d'incidence aux arcs A est une matrice $n \times m$ avec n = |V| et m = |E|. On a $\text{A}_{i,j} = 1$ si il y a un arc j connecté au nœud i et 0 sinon. Dans le cas non orienté suivant l'incidence de l'arc, la valeur prise peut-être 1 ou -1 pour indiquer le sens sortant ou entrant de l'arc.

Le flot maximum est égal à la coupe minimale entre s et d, et, par conséquent, il représente également la K edge-connectivity entre s et d dans G :

$$k_{\text{G}}(s, d) = \max_{f} f(u_r) \tag{4.22}$$

En utilisant une énumération de toutes les paires de nœuds du graphe, nous pouvons exprimer la arc-connectivité moyenne du graphe k(G) comme la connectivité moyenne calculée sur chaque paire de nœuds du graphe:

$$\bar{k}(\text{G}) = \sum_{(i,j),\ i<j} \frac{k_{\text{G}}(i, j)}{\binom{m}{2}} \tag{4.23}$$

Posons $f_{ij} \in \mathbb{R}^n$ le flot relatif à x_i et x_j, incluant l'arc virtuel de retour entre les deux nœuds. En utilisant les équations (4.21) et (4.23) on obtient:

$$\begin{aligned} \underset{f_{ij}}{\text{maximize}} \quad & \frac{\sum_{(i,j),\, i<j} f_{ij}(x_i, x_j)}{\binom{m}{2}} \\ \text{subject to} \quad & \\ & \forall (i,j),\ i < j, \mathrm{A}.f_{ij} = 0 \end{aligned} \tag{4.24}$$

Le problème linéaire intermédiaire suivant produit le sous-graphe (i.e. le groupe) $\mathrm{G}_k(\mathrm{V}_k, \mathrm{E}_k)$ de nombre de nœuds p (*i.e.* $|\mathrm{V}_k| = p$) qui a la connectivité moyenne la plus élevée.

$$\begin{aligned} \underset{\mathrm{A}_k}{\text{maximize}} \quad & \frac{\max \sum_{(i,j) i<j} f_{ij}^k(x_i, x_j)}{\binom{p}{2}} \\ \text{subject to} \quad & \\ & \forall k, |\mathrm{V}_k| = p \\ & \forall k, (i,j), i < j, \mathrm{A}_k.f_{ij}^k = 0 \end{aligned} \tag{4.25}$$

Où A_k est la matrice d'incidence du sous-graphe G_k, est f_{ij}^k est son flot maximum, et $\binom{p}{2}$ est le nombre de paires de nœuds du sous-graphe.

Nous en déduisons notre modèle final qui s'applique à un nombre variable de groupes, un nombre variable de nœuds dans chaque groupe en incluant la charge induite par ces nœuds dans les contrôleurs.

Notre objectif est de sélectionner les groupes optimaux pour maximiser la connectivité moyenne des groupes et d' équilibrer la charge entre eux, en réduisant l'écart entre le groupe le plus chargé et le moins chargé. Nous exprimons donc l'objectif supplémentaire d'équilibrer la charge entre les groupes.

Nous avons les paramètres supplémentaires suivants:

- un nombre de groupes $k_m \in [k_{min}, k_{max}]$ pour gérer la charge induite par l'ensemble des SOFs du réseau et trouver une configuration avec le meilleur équilibre de charge entre les groupes,

- chaque G_{k_m} indique un groupe avec un nombre arbitraire de nœuds, avec k_m compris entre k_{min} et k_{max},

- p_{k_m} est le nombre de nœuds de G_{k_m},

- $l(\mathrm{G}_{k_m})$ est la charge du sous graphe G_{k_m} associé au groupe avec $l(\mathrm{G}_{k_m}) = \sum_{i \in \mathrm{V}_{k_m}} l_i$, où l_i est la charge du nœud i.

Le problème de programmation linéaire bi-objectifs que nous voulons résoudre est le

suivant:

$$\begin{aligned} &\underset{\cup G_{k_m}}{\text{maximize}} && \frac{\min_{k_p \in [1,k_m]} l(G_{k_p})}{\max_{k_n \in [1,k_m]} l(G_{k_n})} \\ &\underset{\cup G_{k_m}}{\text{maximize}} && \sum_{k=1}^{k_m} \frac{\sum_{(i,j) i<j} f^k_{m_{ij}}(x_i, x_j)}{\binom{p_{k_m}}{2}} \\ &\text{subject to} \\ & && \sum_{i=1}^{k_m} |G_{k_i}| = m \\ & && \forall k_m, (i,j), i<j, A_{k_m}.f^{k_m}_{ij} = 0 \end{aligned} \tag{4.26}$$

De manière optionnelle, nous pouvons rajouter un objectif supplémentaire visant à minimiser le nombre de groupes:

$$\underset{\cup G_{k_m}}{\text{minimize}} \quad k_m \tag{4.27}$$

Analyse de la complexité

Soit S un ensemble de K groupes, c'est-à-dire une famille de sous ensembles de V avec |S| = K. Le nombre de partitions existantes, c'est-à-dire la taille de l'espace de recherche est $\left\{ {N \atop K} \right\}$. Ce nombre est connu comme le nombre de stirling de deuxième espèce et croit exponentiellement en N pour K fixé.

Il convient de noter que ce nombre est beaucoup plus élevé que le nombre de combinaisons de k contrôleurs possibles habituellement indiqué dans les articles sur le placement des contrôleurs qui se contentent de calculer la combinatoire de k contrôleurs parmi N nœuds. Par exemple, avec K = 3 et N = 30: nous avons $\binom{N}{K}$ = 4060 et $\left\{ {N \atop K} \right\}$ = 34314651811530.

D'autre part, notre problème est un problème dual d'un cas particulier de couverture minimale d'un ensemble *minimum set coverage* d'après KARP [1972] où le poids d'une partition de l'ensemble de référence est la connectivité moyenne du sous-graphe correspondant. Par conséquent, notre problème de référence est nécessairement NP-difficile. Un facteur d'approximation de $\log(m+1)$ est donné dans [BAR-ILAN et PELEG, 1991a] pour le problème de couverture minimale.

Description de l'algorithme

La structure de base de l'algorithme utilise les mécanismes de classification, de sélection et d'élitisme de NSGA-II présenté en 2.4.4. DEB et collab. [2002] démontrent les très bonnes performances de NSGA II en particulier quand les objectifs sont peu nombreux et antagonistes.

De plus, selon AGRAMA [2011], c'est un outil efficace pour résoudre les problèmes d'emplacements d'établissements à deux objectifs qui présentent des similitudes avec notre propre problème. Comme la connectivité moyenne et l'équilibre de charge entre groupes sont deux objectifs non corrélés, nous avons décidé d'utiliser cet algorithme comme structure de base.

Le premier objectif que nous poursuivons est de maximiser la moyenne de la connectivité moyenne de tous les groupes.

Le second objectif est de minimiser le déséquilibre de charge entre les groupes.

Comme le cadre NSGA-II a une structure très générique, il est utile de noter qu'il peut aussi être facilement étendu à un troisième objectif ou plus en utilisant NSGA-III [BHESADIYA et collab., 2016] pour chercher à minimiser aussi par exemple le nombre de groupes.

Algorithm 6: MAIN: Evolutionary algorithm

```
1:  INPUT: nbGen, popSize, netGraph, MUTPB, CXPB
    kMin, kMax
2:  OUTPUT: pf

3:  for index = 1 to popSize do
4:      k ⇐ random(kMin..kMax)
5:      population ⇐ randomClustering(k, netGraph)
6:  end for

7:  pf = ∅

8:  for index = 1 to nbGen do
9:      for individual ∈ population do
10:         Evaluate(individual)
11:     end for

12:     for index = 1 to popSize do
13:         parents ⇐ binarySelection(population)
14:         child = cross-over(CXPB, parents)
15:         offSpring ⇐ child
16:     end for

17:     for individual ∈ offSpring do
18:         Mutate(MUTPB, individual)
19:     end for

20:     population ⇐ SelectNSGA2(population ∪ offSpring)
21:     pf ⇐ UpdateParetofront(pf, population)
22: end for
23: return pf
```

La structure de l'algorithme est illustrée avec du pseudo-code dans l'algorithme 6. Les paramètres d'entrée et de sortie sont:

- *nbGen*: le nombre de générations,
- *popSize*: la taille de la population,
- *netGraph*: la structure de données qui représente le graphe du réseau,
- *MUTPB*: le taux de mutation, *CXPB*: le taux de recombinaisons,
- *kMin*, *kMax*: nombre minimal et maximal de groupes acceptables pour une solution.

- *pf*: est la sortie, c'est-à-dire l'ensemble de solutions qui constituent le front de Pareto trouvé à la fin de l'algorithme.

La première étape de l'algorithme, en lignes de 3 à 6, consiste à créer au hasard une population d'individus. Dans notre cas, les individus sont une liste de groupes (*clusters*) composés d'un sous-ensemble de nœuds du graphe. En d'autres termes, c'est une liste de sous-graphes distincts qui couvrent l'ensemble du graphe du réseau. Le nombre de groupes de chaque individu k, peut varier entre $k\mathrm{M}in$ et $k\mathrm{M}ax$.

Ces deux seuils sont estimés à partir de la charge cumulée du plan de contrôle induite par l'ensemble des SOFs du réseau. $k\mathrm{M}in$ doit être assez grand pour disposer d'un nombre suffisant de contrôleurs pour accepter la charge du plan de commande induite par tous les SOFs. D'autre part, $k\mathrm{M}ax$ doit être assez grand pour éviter que les contrôleurs ne soient trop surchargés.

L'opérateur d'évaluation L'opérateur d'évaluation, aux lignes 8 à 11, est illustré dans l'algorithme 7. Il calcule une valeur correspondant à chaque objectif, pour chaque individu. Dans notre cas, le premier objectif est la moyenne de la connectivité moyenne de chaque sous-graphe correspondant à chaque groupe de l'individu. La arc-connectivité moyenne d'un sous-graphe est calculée à l'aide d'une routine classique basée sur l'évaluation du flot maximum entre chaque paire de nœuds et calculée en moyenne sur toutes les paires de nœuds du sous-graphe.

La arc-connectivité moyenne d'un sous-graphe est évaluée à 0 si le sous-graphe a plusieurs composantes connexes, c'est-à-dire qu'il est déconnecté. Cela permet d'accélérer notablement les calculs en supprimant des calculs inutiles.

Le deuxième objectif évalué est le déséquilibre de charge entre le groupe le plus chargé et le groupe le moins chargé d'un individu.

La charge d'un groupe (c'est-à-dire d'un sous graphe) est exprimée comme la somme de la charge du plan de contrôle induite par chaque nœud du groupe. Si la valeur calculée vaut 1, la charge est parfaitement équilibrée entre les différents groupes d'un individu.

L'opérateur de mutation Le mécanisme de l'opérateur de mutation est illustré dans l'algorithme 8 entre les lignes 12 à 14. Il consiste à échanger quelques nœuds entre les groupes des individus. Sur la base du taux de mutation, un certain nombre de nœuds sont extraits aléatoirement du réseau et ré-attribués de la même manière dans les groupes de l'individu courant en utilisant une loi de probabilité uniforme. La proportion de nœuds extraits aléatoirement et réalloués est définie par le paramètre *MUTPB*.

L'opérateur de recombinaison L'opérateur de recombinaison, en lignes de 15 à 19, est optionnel et est illustré dans l'algorithme 9.

Le mécanisme de l'opérateur de recombinaison a été défini par nous-même et inclut un processus d'optimisation locale dont l'objectif est d'accélérer la convergence de l'algorithme en réduisant le nombre de générations. Nous pouvons le décrire comme un mécanisme Lamarckiste où les individus sont en quelque sorte améliorés tout au long de leur vie [HOLZINGER et collab., 2014].

Pour produire un enfant, deux parents sont choisis au hasard en utilisant le processus habituel de sélection par tournois binaires. Le taux de croisement CXPB fixe la probabilité que le croisement entre deux parents soit réalisé.

Ensuite, l'ensemble des groupes provenant des deux parents sont regroupés dans l'ensemble de groupes noté *clusters* à la ligne 3. Puis, un nombre aléatoire k de groupes

Algorithm 7: OPERATOR: Evaluation

1: **INPUT:** $individual, netGraph$

2: **for** $eachCluster \in individual$ **do**
3: **for** $eachPairOfNode \in eachCluster$ **do**
4: $clustAvConnectivity \Leftarrow edgeConnectivity(eachPairOfNode)$
5: **end for**
6: $clustAvConnectivity \Leftarrow \frac{clustAvConnectivity}{\binom{|clustAvConnectivity|}{2}}$
7: $indConnectivity = indConnectivity + clustAvConnectivity$
8: **end for**
9: $indConnectivity \Leftarrow \frac{indConnectivity}{numberOfClustInd}$

10: **for** $eachCluster \in individual$ **do**
11: **for** $eachNode \in cluster$ **do**
12: $clustLoad = clustLoad + load(eachNode)$
13: **end for**
14: $indLoad = indLoad + clustLoad$
15: **end for**
16: $indUnbalance = maxClusterLoad(indLoad)/minClusterLoad(indLoad)$
17: $individual \cdot unbalance \Leftarrow indUnbalance$
18: $individual \cdot connectivity \Leftarrow indConnectivity$

19: **return** $individual$

Algorithm 8: OPERATOR: Mutation

INPUT: $individual$, MUTPB

$nodes \Leftarrow random(individual, \text{MUTPB})$

for $eachNode \in nodes$ **do**
$reallocateRandomly(node, individual)$
end for
return $individual$

entre $k\text{M}in$ et $k\text{M}ax$ pour l'enfant est sélectionné dans cet ensemble, à la ligne 4. Cette sélection est réalisée au hasard ou à l'aide d'une stratégie déterministe. Nous avons imaginé et testé différentes stratégies:

- Les k groupes sont extraits aléatoirement de l'ensemble *clusters*,
- Les groupes sont sélectionnés en utilisant une procédure de sélection qui choisie les k groupes qui ont la meilleure connectivité moyenne, améliorant par conséquent la fitness de connectivité de l'individu,
- Les groupes sont sélectionnés en utilisant une procédure de sélection qui choisit parmi les groupes les k meilleurs du point de vue de l'équilibre de charge $\frac{load}{nbNodes}$.

Algorithm 9: OPERATOR: Cross-over

1: **INPUT:** $parents$, CXPB, $allNodes, stratSel$
2: $kMin, kMax, stratRealloc$

3: $clusters \Leftarrow clusters(parents)$
4: $child \Leftarrow selectClusters(clusters, kMin, kMax, stratSel)$

5: $redundantNodes \Leftarrow \cap(clusters \in child)$
6: $childNodes \Leftarrow \cup(clusters \in child)$
7: $nodesBag = redundantNodes \cup (allNodes - childNodes)$

8: **for** $eachNode \in nodesBag$ **do**
9: $\quad reallocateNodes(eachNode, child, stratRealloc)$
10: **end for**

11: **return** $child$

Ce processus offre une première occasion d'améliorer localement les individus à chaque génération par rapport à l'un des objectifs. Mais nous mettons aussi en œuvre un second processus d'optimisation locale.

Dans l'ensemble de groupes sélectionnés, il y a naturellement des nœuds redondants et des nœuds manquants. Ils sont stockés en mémoire en ligne 7. Il y a ensuite réaffectation dans les groupes de l'enfant dans la boucle de 8 à 10 en utilisant différentes stratégies visant à améliorer un ou plusieurs objectif. Un exemple est illustré dans l'algorithme 10 où la fonction de réaffectation vise à sélectionner comme destination pour le nœuds, le cluster dont la connectivité sera maximisée. D'autres stratégies ont été expérimentées:

- Une stratégie qui vise à maximiser la connectivité moyenne de chaque individu,
- Une stratégie visant à maximiser la variation entre la connectivité moyenne avant allocation et après allocation des nœuds,
- Une stratégie visant à maximiser l'adaptation d'équilibre de charge de l'individu,
- Une stratégie visant à maximiser la variation entre l'équilibre de charge avant et après allocation.

Complexité

La complexité de l'algorithme NSGA II est conditionnée par le mécanisme de classification basé sur la dominance de l'algorithme NSGA II. Celle-ci est conditionnée d'abord par:

$$O(M \cdot popSize^2)$$

où M représente le nombre d'objectifs.

Pour exprimer la complexité globale de l'algorithme il faut aussi prendre en compte, le nombre de générations *nbGen*, et la complexité de calcul des opérations additionnelles qui interviennent à chaque génération:

Algorithm 10: OPERATOR: Reallocation

1: **INPUT:** $node, child, stratRealloc$

2: $max = 0, clusterMax = \{\}$
3: **for** $eachCluster \in child$ **do**
4: $delta = averageConnect(eachCluster \cup node) - averageConnect(eachCluster)$
5: **if** $delta \geq max$ **then**
6: $deltaMax \Leftarrow delta$
7: $clusterMax \Leftarrow eachCluster$
8: **end if**
9: $clusterMax \Leftarrow clusterMax \cup node$
10: **end for**

11: **return** $child$

- Le processus d'évaluation de chaque individu,
- l'opérateur de mutation,
- l'opérateur de recombinaison,
- le processus de sélection.

Complexité du processus d'évaluation des individus La complexité du processus d'évaluation est bornée par la complexité du calcul de la connectivité entre deux nœuds qui est en $O(|E|^2 \cdot |V|)$ pour l'algorithme de EDMONDS et KARP [1972] que nous utilisons. Il faut y ajouter le calcul de la charge globale du graphe du réseau en $O(|V|)$. Par conséquent nous avons une complexité de calcul en $O(|E|^2 \cdot |V|^3 + |V|)$ pour un individu dans le pire cas.

Complexité de l'opérateur de mutation La complexité de l'opérateur de mutation est bornée par le nombre de nœuds du graphe en $O(|V|)$. L'opération consiste en effet seulement à échanger aléatoirement quelques nœuds entre les groupes.

Complexité de l'opérateur de recombinaison La complexité de l'opérateur de recombinaison dépend de la stratégie de croisement choisie puisqu'il inclut un mécanisme d'optimisation local. Si le mécanisme est basé sur le calcul de la connectivité moyenne de chaque groupe et en considérant le groupe de taille maximale qui peut être dans le pire cas, le graphe lui même, la complexité est bornée en $O(|E|^2 \cdot |V|^3)$ fois le nombre de groupes , pour l'étape de sélection des groupes et en $O(|E|^2 \cdot |V|^3)$ fois le nombre de nœuds pour l'étape de ré-allocation des nœuds.

Complexité globale Pour résumer notre analyse de complexité, sans opérateur de recombinaison la complexité de calcul est bornée en:

$$O(nbGen \cdot (M \cdot popSize^2 + popSize \cdot |E|^2 \cdot |V|^3)).$$

Elle est acceptable même si le réseau contient quelques centaines de nœuds en considérant qu'il s'agit du pire cas.

Toutefois, si la stratégie adoptée pour l'opérateur de recombinaison est basée sur l'optimisation de la connectivité moyenne, la complexité de calcul est guidée par cet objectif et est bornée en:

$$\mathrm{O}(nbGen \cdot (\mathrm{M} \cdot popSize^2 + popSize \cdot (|\mathrm{E}|^2 \cdot |\mathrm{V}|^3 + |\mathrm{E}|^2 \cdot |\mathrm{V}|^4))$$

et le temps de calcul peut devenir prohibitif même pour des réseaux de taille moyenne. Dans ce cas, il peut être utile et même indispensable de chercher une méthode d'estimation de la connectivité moyenne, plutôt que d'effectuer un calcul exact. Pour cela nous avons envisagé deux approches.

Estimation de la connectivité moyenne

Estimation analytique Pour que l'opérateur de recombinaison reste utilisable tout en maintenant un temps de calcul acceptable, nous avons introduit une estimation analytique de la connectivité moyenne proposée par [BEINEKE et collab., 2002a] et utilisable à l'intérieur de l'opérateur de recombinaison.

Cette stratégie est acceptable dans le contexte d'un algorithme évolutionnaire où des mécanismes aléatoires peuvent être introduits à différents stades de l'algorithme pour maintenir une certaine variabilité dans la population des solutions.

Ainsi, l'estimation de la connectivité de chaque groupe fournit un moyen d'améliorer la solution courante, en même temps que l'imprécision de l'approximation introduit un certain caractère aléatoire dans les boucles de l'algorithme. Ce qui peut également être utile afin d'éviter une convergence trop précoce.

Nous utilisons l'approximation analytique grossière suivante pour la arc-connectivité moyenne où n est le nombre de sommets et $\bar{d}$ le degré moyen du graphe G :

$$\bar{d} \cdot \frac{\bar{d}}{n-1} \leq \bar{k}_{\mathrm{G}}(n, \bar{d}) \leq \bar{d} \tag{4.28}$$

Et nous sélectionnons comme valeur, la valeur moyenne entre les deux bornes. Cette estimation analytique de la connectivité moyenne peut être calculée très rapidement, ce qui permet à l'algorithme de maintenir un temps de calcul proche de celui obtenu avec l'opérateur de mutation seul.

Estimation par une méthode de Monte Carlo Nous avons expérimenté une autre manière d'estimer la connectivité moyenne d'un sous-graphe, en utilisant une estimation de Monte Carlo. L'idée est de choisir aléatoirement un certain nombre de nœuds par exemple 10% ou 20% et de calculer la connectivité moyenne de ce sous-ensemble de nœuds en considérant qu'elle constitue une bonne approximation de la connectivité moyenne. Nous avons pu vérifier que cette méthode ne donne pas de meilleurs résultats que la méthode analytique décrite précédemment tout en étant plus coûteuse en temps de calcul.

Evaluation de la performance de l'algorithme

Après avoir décrit en détail notre solution, nous présentons maintenant son évaluation. Cette dernière repose principalement sur des comparaisons de la version mutation pure de l'algorithme, sans étape de recombinaison, fonctionnant sur un nombre pertinent

de générations avec notre modèle de référence exécuté sur un solveur ILP d'une part et d'autre part avec la version de l'algorithme qui inclut une étape de recombinaison privilégiant soit la connectivité soit la charge des groupes.

Nous utilisons pour notre évaluation des instances de réseau extraites de la base de données zoo-topologie décrite en 4.3.9 complétées de manière à n'avoir que des graphes connectés. Une charge de plan de contrôle a été générée aléatoirement et attachée aux nœuds des instances de réseau.

Analyse qualitative de l'algorithme Nous avons également évalué l'algorithme sur certaines configurations de petits réseaux pertinents en utilisant la version mutation pure sans opérateur de recombinaison. La figure 4.35 illustre une configuration réseau avec trois groupes simples. Chaque groupe a une connectivité moyenne élevée par rapport à celle du réseau et ils sont reliés entre eux par un seul arc. Chaque nœud induit la même charge pour un contrôleur, c'est-à-dire que la charge est homogène à l'intérieur du réseau. Dans ce scénario, l'algorithme produit dans un très faible nombre de générations (autour de 10), l'individu qui est la meilleure configuration de trois groupes pour obtenir la connectivité moyenne la plus élevée: $\{1,2,3,4\},\{5,6,7,8\},\{9,10,11,12\}$ avec $\sum_i G_{k_i} = 6$.

Le résultat est cohérent avec le résultat obtenu à l'aide de l'implémentation du modèle en Minizinc [NETHERCOTE et collab., 2007a] sur le solveur Geocod. Cela nous permet aussi de valider notre modèle ILP.

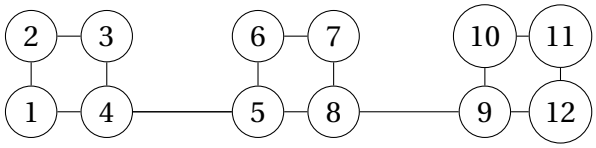


Figure 4.35 – Exemple du petit réseau

Dans la figure 4.36, le réseau est organisé en trois centres. Dans ce cas, l'arborescence du réseau rend l'analyse de connectivité moyenne inefficace car la connectivité moyenne d'un groupe qui est un arbre vaut toujours 1.

Dans ce scénario, l'analyse de charge devient le seul critère pour distinguer les groupes. Comme dans l'évaluation précédente, la charge est homogène. Dans ce cas, si on fixe le nombre de groupes à 3 dans l'algorithme, celui-ci divise le réseau en trois groupes, chaque groupe correspondant à un centre : $\{1,2,3,4\},\{5,6,7,8\},\{9,10,11,12\}$ et $\sum_i G_{k_i} = 3$.

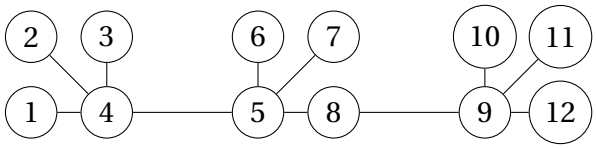


Figure 4.36 – Exemple du réseau en rayon

Dans la figure 4.37 nous avons plusieurs groupes avec des valeurs de connectivité moyenne différentes. La version mutation seule de l'algorithme est utilisée avec un faible nombre de générations égal à 20 en raison de la petite taille de l'instance réseau.

Lors de la sélection d'un nombre raisonnable de groupes, variant de 2 à 6, la meilleure solution est obtenue pour trois groupes:

$\{1,2,3,4,\}\{5,6,7,8\}\{9,10,11,12\}$, où la connectivité moyenne vaut: 3 pour le premier groupe, 2.16 pour le second et 2 pour le troisième groupe. La connectivité moyenne des trois groupes réunis vaut 2.38.

Ces résultats sont cohérents avec ceux fournis par une implémentation de notre modèle linéaire en nombres entiers à l'aide du solveur Minizinc et permettent aussi de vérifier la calibration du solveur.

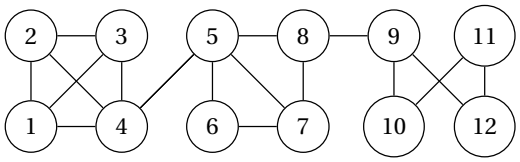


Figure 4.37 – Un autre exemple de petit réseau

Performances par rapport à la solution de référence Afin d'avoir une solution de référence, nous avons comparé l'approche mutation pure avec les résultats optimaux obtenus avec le solveur Geocode [TACK, 2009] et le langage de modélisation minizinc [NETHERCOTE et collab., 2007b] pour exprimer notre modèle de programmation linéaire en nombres entiers décrit en 4.4.3.

Pour comparer les résultats obtenus avec la version de l'algorithme à mutation seule, nous avons exprimé le déséquilibre de charge et le nombre de groupes d'une solution sélectionnée dans le front de Pareto fourni par l'AE comme deux contraintes dans le solveur. Nous avons ensuite comparé la solution fournie par le solveur et la valeur moyenne de connectivité de certaines solutions proposées sur le front de Pareto pour un nombre de contrôleurs donné.

Le temps d'exécution prohibitif nécessaire pour résoudre ce problème d'optimisation pour le solveur nous a limité aux réseaux avec un petit nombre de nœuds et d'arcs, environ 20 à 30 nœuds ou arcs. La figure 4.3 représente le décalage d'équilibre de charge sur les groupes obtenus en utilisant la solution proposée, avec la solution optimale fournie par le solveur. Les résultats obtenus sont présentés dans le diagramme 4.38.

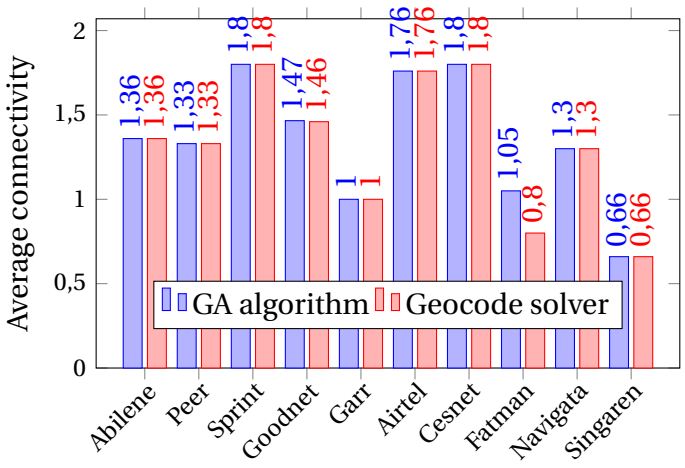


Figure 4.38 – Comparaison avec Minizinc

Les résultats obtenus avec le solveur sont la plupart du temps identiques ou très proches de ceux fournis par l'AE. Parfois, il y a un décalage (cas du réseau "Fatman" de la base de données Zoo-topology) en raison du temps limité prévu pour résoudre le problème et trouver la solution optimale c'est-à-dire plusieurs heures. Le fait que le nombre de groupes à résoudre est supérieur à 3 dans ce cas, élargit considérablement l'espace de recherche.

Nous présentons les caractéristiques pertinentes de ces réseaux (nombre de nœuds et d'arcs et nombre de groupes pour le calcul) dans le tableau 4.3. Pour ces petits réseaux, nous vérifions que la version à mutation seule est capable de fournir la meilleure solution en quelques secondes avec un faible nombre de générations (environ 20).

Networks features			
Network	Nb of nodes	nb of Edges	nb of clusters
Abilene	11	13	3
Peer	17	19	3
Sprint	12	18	2
Goodnet	18	18	2
Garr	16	23	3
Airtel	16	26	3
Cesnet	12	12	3
Fatman	17	21	4
Navigata	13	17	2
Singaren	11	10	3

Table 4.3 – Caractéristiques des réseaux

Comparaison avec l'algorithme Kmean++ Pour compléter l'évaluation de l'algorithme et sa capacité à fournir de bonnes solutions nous avons comparé ses performances sur certains réseaux de taille moyenne (*i.e.* en utilisant l'implémentation d'un algorithme de groupage K-moyennes++ [ARTHUR et VASSILVITSKII, 2007] et un scénario restreint, où nous fixons le nombre de groupes avec une valeur arbitraire, ($k = 3$) comme il est nécessaire de le faire avec K-moyennes.

D'autre part, l'algorithme utilise comme entrée la matrice de connectivité du graphe du réseau pour mesurer la similarité entre les nœuds. De plus, la charge de contrôle induite par chaque nœud est fixée à la même valeur. Ensuite, nous exécutons l'AE sur 100 générations, ce qui est un nombre raisonnable pour obtenir de bons résultats avec les instances réseau de cette taille. La différence entre les résultats pour la connectivité moyenne et la charge est affichée dans la Figure 4.39. La figure montre que l'AE donne de meilleurs résultats même dans ce scénario simpliste. Les points de connectivité moyens et leurs valeurs de déséquilibre de charge obtenus avec l'AE sont toujours meilleurs que les points obtenus avec l'algorithme K-moyennes. Même si parfois la connectivité moyenne trouvée est la même, le déséquilibre de charge est toujours plus mauvais avec K-moyennes.

Le nombre de nœuds et d'arcs de chaque réseau varie entre 20 et 70 dans cette expérience et sont présentés dans le tableau 4.4.

Convergence

Version mutation seule La figure 4.40 illustre un exemple de front de Pareto obtenu sur un réseau typique de taille moyenne avec 40 nœuds et 60 arcs extraits de la base de données zoo-topology. La courbe en vert montre un front de Pareto obtenu après 50 générations avec la version mutation pure de l'algorithme et un nombre de groupes variant de 2 à 6. Le taux de mutation MUTPB est fixé à 0.05 et le taux de mutation CXPB est

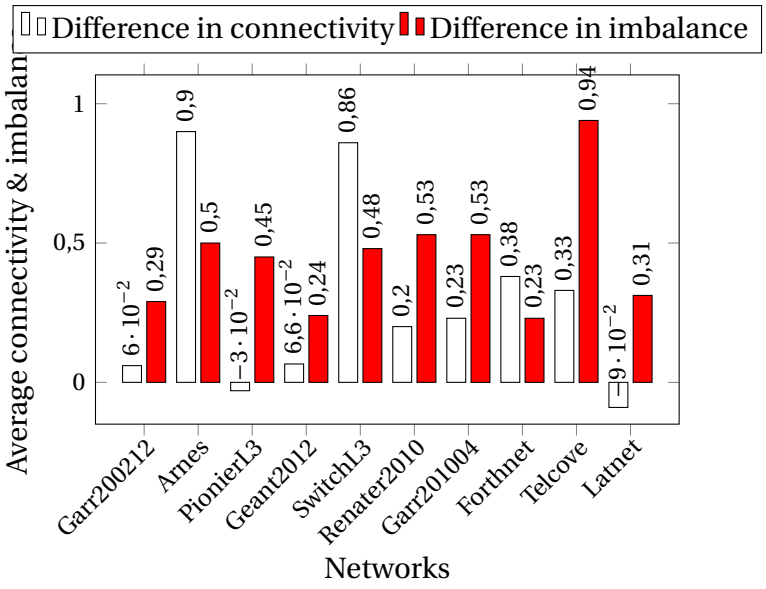


Figure 4.39 – Comparaison avec Kmean++

Networks features: nodes and edges		
Network	Nb of nodes	nb of Edges
Garr200212	28	27
Arnes	34	45
PionnierL3	38	45
Geant2012	40	60
SwitchL3	42	62
Renater2010	43	55
Garr201004	54	68
Forthnet	62	62
AsnetAM	65	77
Telcove	72	73
Latnet	74	69

Table 4.4 – Nombre de nœuds et d'arcs des réseaux testés

fixé à 0.9, la taille de la population est de 200 individus. La valeur optimale pour l'équilibre de charge est de 1, la connectivité moyenne dépend des topologies des réseaux. Pour des valeurs de connectivité moyenne faibles, les groupes sont parfaitement équilibrés tandis que pour des valeurs de connectivité élevée, le déséquilibre augmente.

La figure 4.41 présente sous forme de courbe, la convergence au fil des générations de l'algorithme à mutation seule s'exécutant sur le réseau de taille moyenne objet du test précédent.

Nous itérons 50 fois l'algorithme pour fournir une valeur moyenne pertinente de chaque valeur calculée au fil des générations.

La métrique illustrée dans cette courbe est l'indicateur d'hyper-volume décrit par [AUGER et collab., 2012] du front de Pareto. La courbe intègre une visualisation de l'intervalle de confiance pour chaque point au fil des générations.

L'indicateur d'hyper-volume est calculé par rapport à un point de référence.

Nous avons choisi celui qui correspond à la meilleure valeur d'équilibre de charge c'est-à-dire 1.0 et à la valeur de connectivité moyenne la plus élevée, arrondie à une valeur

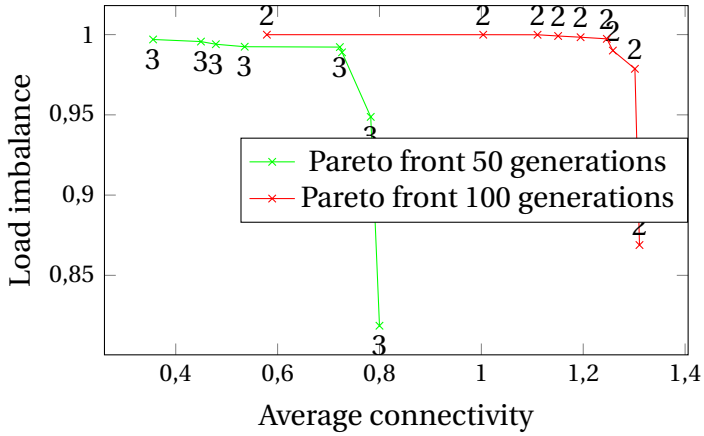


Figure 4.40 – Réseau de taille moyenne: 40 nodes, 60 edges

entière parmi tous les points des différents fronts de Pareto de chaque génération.

L'hyper-volume correspond à la surface entre le front de Pareto et le rectangle défini par le point de référence et le point d'origine du diagramme.

Plus l'indicateur d'hyper-volume est bas, meilleures sont les solutions apportées par le front de Pareto. On constate qu'il n'y a pas d'amélioration significative des trois critères au-delà de 150 générations.

Nous présentons ensuite deux courbes du même algorithme correspondant à deux métriques: La connectivité moyenne maximale obtenue parmi tous les points du front de Pareto à chaque génération: 4.42 ainsi que la valeur du déséquilibre de charge pour ce même point 4.43.

Option avec recombinaison Nous avons expérimenté plusieurs stratégies de recombinaison basées sur l'optimisation locale de l'équilibre de charge entre les groupes ou/et l'optimisation de la connectivité moyenne pour la sélection des groupes (fonction selectClusters) et pour la réaffectation des nœuds (fonction reallocate) dans la routine de recombinaison 9.

Pour la sélection des clusters:

A) Une stratégie obtenue en choisissant au hasard les groupes des deux parents pour élaborer un enfant,

B) Une stratégie de sélection des meilleurs groupes des parents basée sur la connectivité moyenne,

C) Une stratégie de sélection des meilleurs groupes des parents basée sur l'équilibre de charge.

Pour la réallocation des nœuds dans les groupes:

1. Une stratégie basée sur l'optimisation de la connectivité moyenne des groupes,
2. Une stratégie basée sur l'optimisation de l'équilibre de charge entre groupes,
3. Une stratégie basée sur la variation de connectivité moyenne avant et après allocation des nœuds,
4. Une stratégie basée sur la variation d'équilibre de charge avant et après allocation des nœuds.

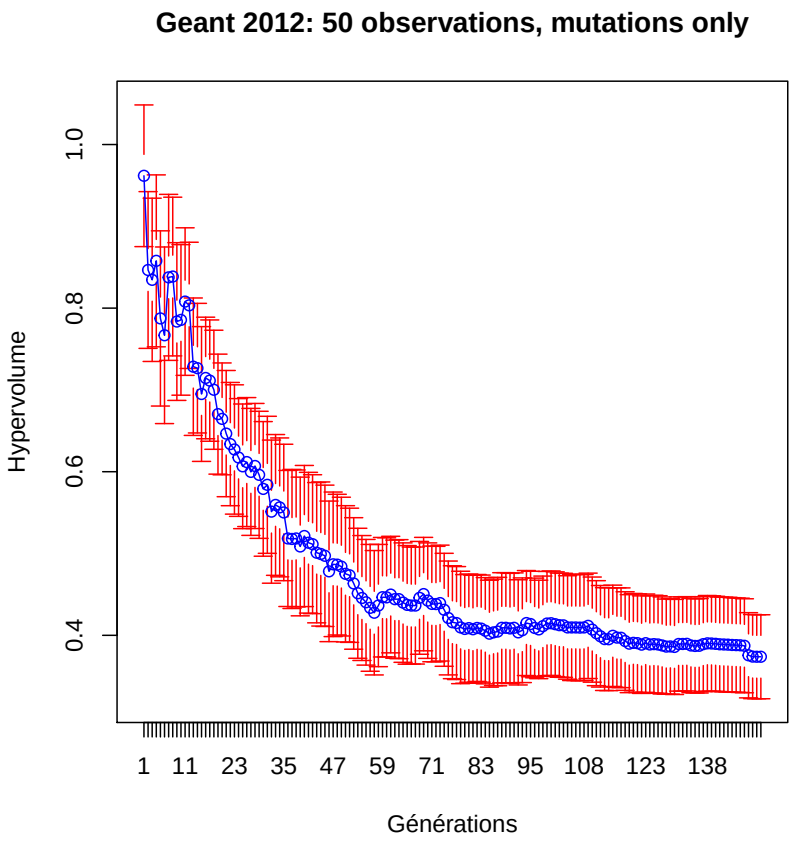


Figure 4.41 – Moyenne de l'hypervolume et intervalle de confiance 95%

La stratégie qui donne les meilleurs résultats sélectionne les groupes au hasard et réalloue les nœuds de manière à maximiser la variation de la connectivité moyenne avant et après allocation.

Un résultat moyenné sur 50 itérations de l'algorithme est illustré dans la figure 4.44 pour un exemple d'instance de réseau. Il montre une amélioration significative de l'indicateur d'hyper-volume par rapport à la version mutation seule présentée en 4.41.

Ceci correspond graphiquement à un déplacement du front de Pareto vers la direction du point de référence qui se trouve dans le coin supérieur droit. On voit aussi une accélération de la convergence (en terme de nombre de générations) par rapport à l'approche mutation seule. On peut aussi remarquer que la courbe de connectivité maximum est très similaire à celle du scénario mutation seule, ainsi que la courbe d'équilibre de charge.

L'amélioration due au croisement se situe donc plutôt dans la qualité de répartition des points le long du Front de Pareto, ou la découverte de points supplémentaires.

Ensuite nous avons comparé les résultats obtenus sur un ensemble de 10 réseaux de taille moyenne (entre 30 et 70 nœuds) avec les deux versions de l'algorithme, c'est à dire la version mutation seule, puis celle qui utilise notre stratégie A3 de réattribution des nœuds basée sur la connectivité dans le mécanisme de recombinaison.

Cette version de l'algorithme calcule la valeur exacte de la connectivité moyenne pour les routines de sélection et de réaffectation utilisées dans l'opérateur de recombinaison. Ce qui implique un temps de calcul assez lourd. La figure 4.47 présente le gain moyen de l'hypervolume obtenu pour le cas A3 comparé à la version mutation pure pour les réseaux

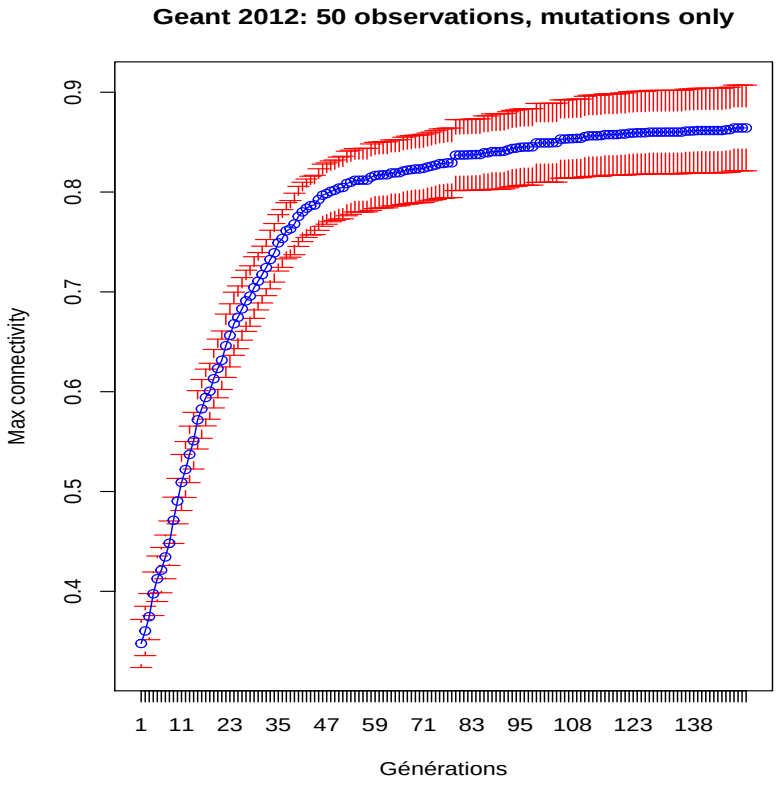


Figure 4.42 – Moyenne de la connectivité maximale et intervalle de confiance 95%

testés.

On constate que l'amélioration de l'hyper-volume peut atteindre plus de 20 % en moyenne sur certains réseaux. Ce gain doit être mis en regard de l'allongement du temps de calcul dû au calcul de la connectivité moyenne pour ré-attribuer les nœuds aux groupes (clusters). Il faut noter cependant que sur un des réseaux, l'hypervolume est légèrement plus dégradé.

Nous avons ensuite modifié dans la stratégie de réallocation de la routine de recombinaison, le calcul de la connectivité moyenne en utilisant une estimation analytique très rapide, pour réduire le problème de temps de calcul.

Nous avons ensuite comparé les résultats obtenus sur un ensemble de 63 réseaux de tailles différentes de 11 à 193 nœuds entre les deux versions de l'algorithme, la version mutation pure, avec celle qui utilise l'algorithme de recombinaison. La stratégie de ré-attribution basée sur la connectivité A3 est la même que précédemment. Le calcul est effectué sur 140 générations et moyenné sur 50 itérations. Le diagramme de comparaison 4.48 montre une nette amélioration de l'hypervolume sur la majorité des réseaux.

La figure 4.49 présente le gain moyen par rapport à la métrique de connectivité maximale sur les points du front de Pareto obtenue pour le cas A3 par rapport à la version mutation pure. Nous ne présentons pas la figure illustrant la métrique d'équilibre de charge pour laquelle nous avons constaté que la différence est pratiquement nulle pour l'ensemble des réseaux.

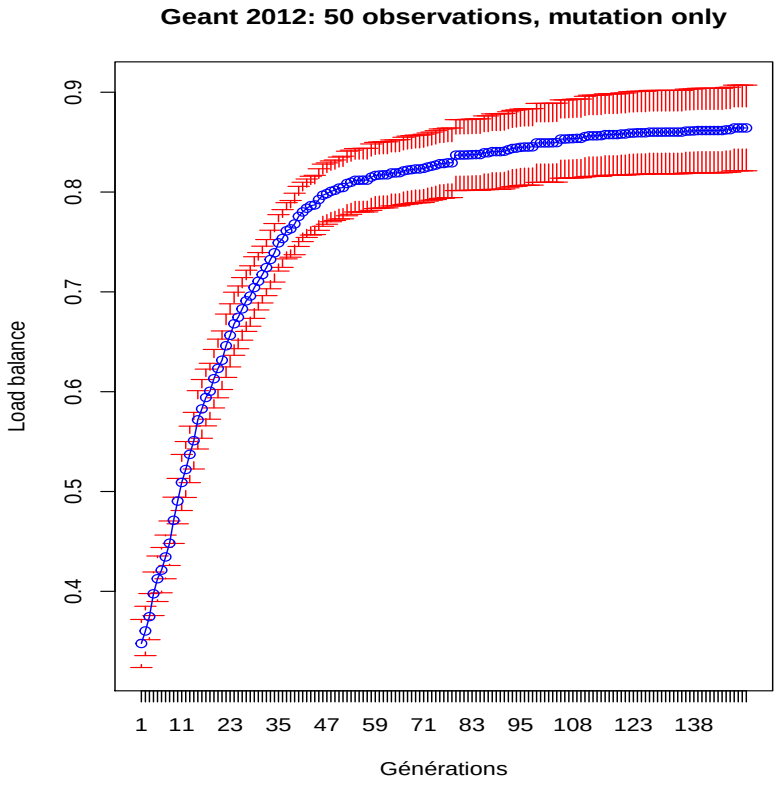


Figure 4.43 – Moyenne de l'équilibre de charge et intervalle de confiance 95%

Placement dynamique

Le but de cette section est de démontrer que moyennant certaines adaptations, cet algorithme peut être utilisé dans un scénario dynamique où la topologie varie dans le temps en raison de défaillances des liens ou des nœuds ou encore en raison de goulots d'étranglement du trafic qui nécessitent des évolutions de la topologie. Des travaux comme celui réalisé par [OTOKURA et collab., 2016] démontrent clairement le gain que l'on peut attendre d'une telle approche. Le principe de base est basé sur l'observation que dans le contexte du vivant des environnements qui varient peuvent accélérer l'évolution [KASHTAN et collab., 2007]. Ceci peut être interprété comme le fait qu'introduire une certaine variation dans l'environnement équivaut à introduire un opérateur stochastique un peu comme le fait dans la structure habituelle de l'algorithme l' opérateur de mutation.

Pour réaliser cette idée, le principe est de garder en mémoire la dernière population calculée qui fournit la solution courante du problème qui a été retenue dans la population. Lorsque de nouvelles conditions réseau apparaissent, par exemple en cas de défaillance de nœuds ou de liens, il est alors nécessaire de recalculer la configuration des contrôleurs. Au lieu de redémarrer l'algorithme à partir d'une population choisie au hasard avec le nombre habituel de générations nécessaires pour obtenir une convergence, l'algorithme démarre de la population courante et la fait évoluer en un nombre réduit de générations en exécutant les tâches de calcul sur la nouvelle topologie. Introduire une variation dans la topologie équivaut à introduire à nouveau un processus stochastique dans la population et permet de rendre l'algorithme à même de converger à nouveau en un plus petit nombre de générations vers une nouvelle solution optimale. Le processus est illustré dans la figure 4.50.

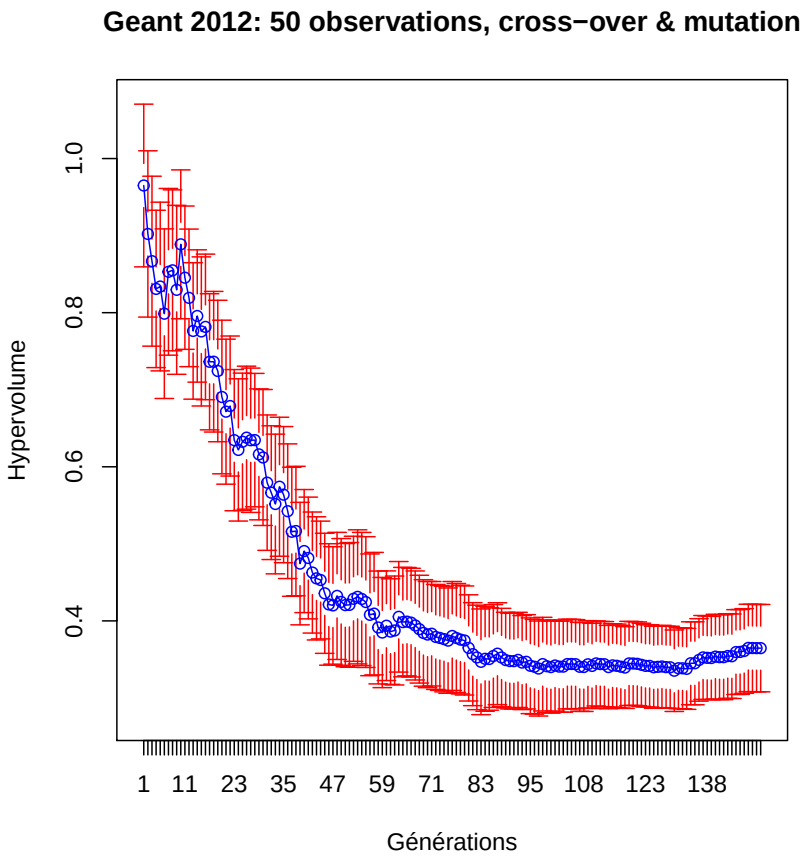


Figure 4.44 – Moyenne de l'hypervolume et intervalle de confiance 95%

Pour expérimenter cette approche, nous sommes partis de la topologie du réseau Cernet composée de 41 nœuds et 57 liens. Nous avons ensuite exécuté l'algorithme standard initié avec une population aléatoire sur cette topologie et nous avons gardé en mémoire la population résultante. Ensuite, nous avons enlevé successivement au hasard, à 5 reprises de 1 à 3 liens pour obtenir 15 topologies dérivées qui diffèrent de 1 à 3 liens de la topologie de référence. Ces modifications correspondraient à des scénarios de changement de topologies induits par des pannes. Nous avons ensuite comparé la convergence de l'algorithme standard sur ces topologies avec la convergence obtenue en exécutant l'algorithme démarré à partir de la population conservée en mémoire. On note qu'un faible nombre de générations entre 20 et 40 selon l'ampleur de la variation provoquée c'est-à-dire le nombre d' arcs enlevés, permet d'obtenir une valeur proche de l'optimal de l'hypervolume. Les résultats sont résumés dans le diagramme 4.51. Le pourcentage négatif indique une dégradation par rapport à l'hypervolume de référence, les résultats positifs indiquent une amélioration par rapport à l'hypervolume de référence. En moyenne, le meilleur résultat est obtenu lorsque deux arcs sont supprimés, avec une amélioration moyenne sur les cinq topologies d'environ 5% sur l'hypervolume.

Le tableau 4.5 montre les valeurs moyennes du gain pour chaque plage de topologies, et la variance obtenue sur les 5 expériences.

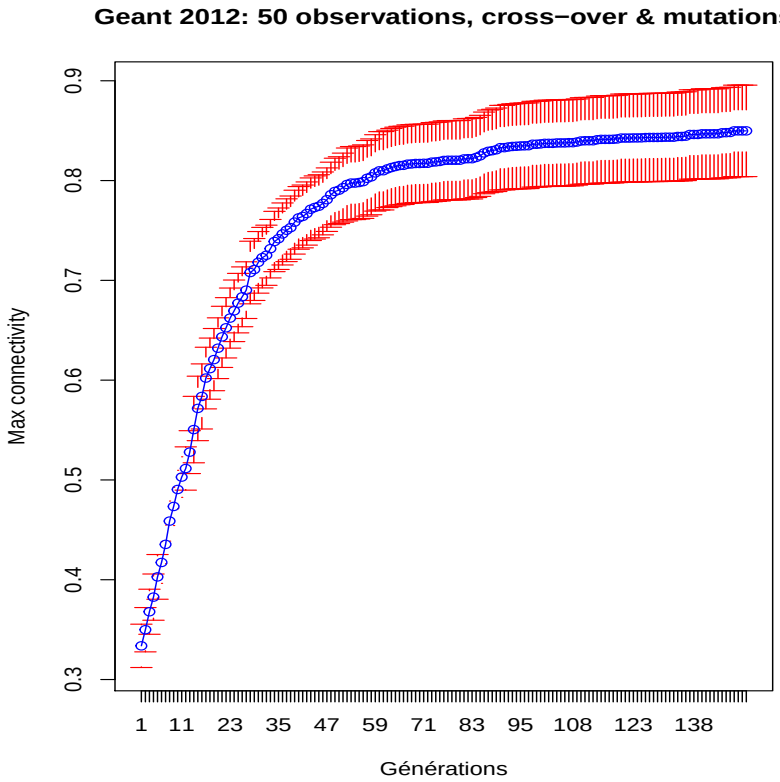


Figure 4.45 – Moyenne de la connectivité maximale et intervalle de confiance 95%

Nbre d'arcs manquants	Gain moyen en%	Variance en%
1	2,18%	-0,12%
2	5,38%	0,27%
3	0,76%	0,92%

Table 4.5 – Gain moyen et variance en % pour chaque cas

4.4.4 Conclusion

Nous avons proposé dans ce travail une approche basée sur les AEMO pour le placement de contrôleurs, qui permet d'obtenir des solutions proches de l'optimal et même optimales pour de petites instances de réseaux. Nous avons validé cela en effectuant des comparaisons avec les solutions fournies par un solveur ILP.

Par ailleurs, pour des instances réseau moyennes ou même grandes de plusieurs centaines de nœuds, l'algorithme est capable de fournir de bonnes solutions en un petit nombre de générations, environ 150. De plus, un opérateur de recombinaison peut également être introduit pour améliorer l'algorithme de manière à réduire le nombre de générations ou à améliorer les résultats. Nous avons proposé une stratégie de croisement qui inclut un processus d'optimisation local basé sur une estimation analytique de la connectivité moyenne. Grâce à cette estimation, le délai de calcul pour fournir des solutions reste pratiquement le même qu'avec la version mutation seule alors que l'algorithme est capable de fournir de meilleures solutions dans le même nombre de générations. Nous avons montré également que l'algorithme est capable de gérer des scénarios dynamiques impliquant des changements de topologies et de réagir rapidement pour fournir de nouvelles

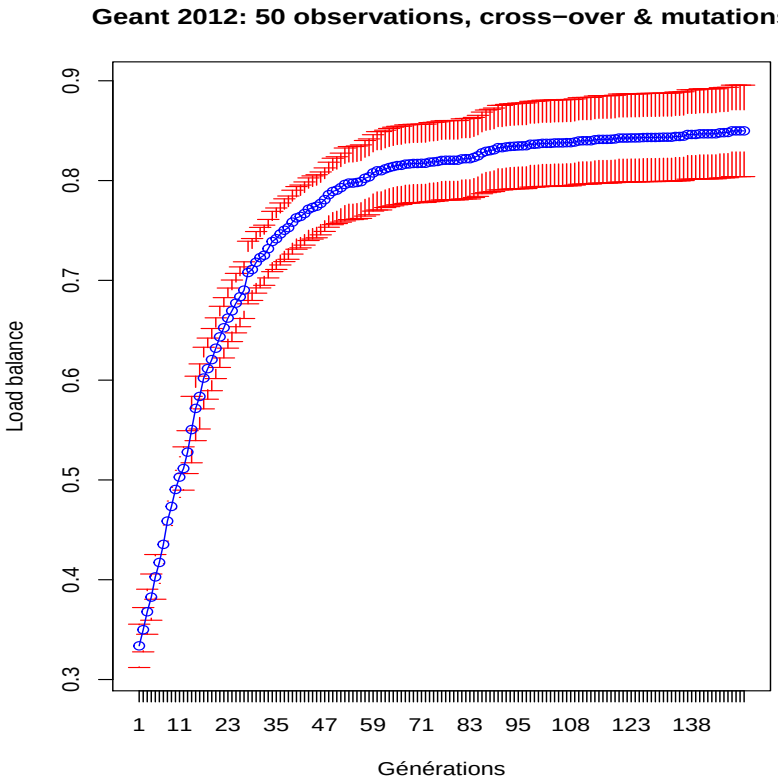


Figure 4.46 – Moyenne de l'équilibre de charge et intervalle de confiance 95%

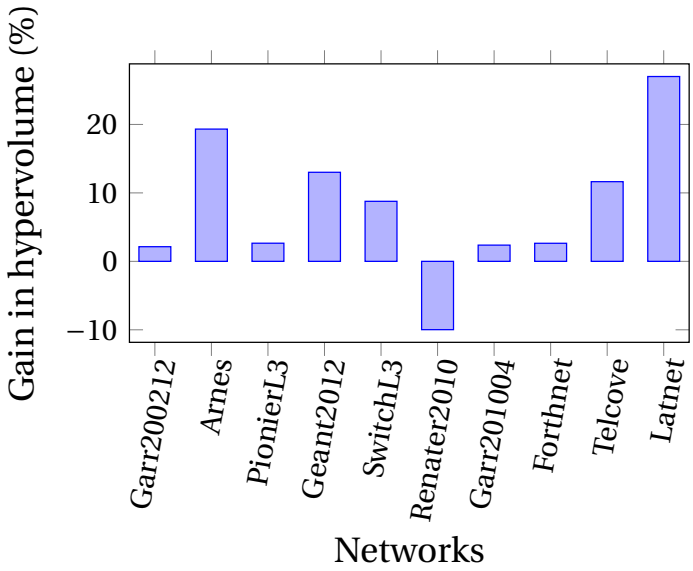


Figure 4.47 – Réseaux de tailles moyennes: 40 nœuds, 60 arcs

solutions de placement de contrôleurs.

La structure de cet algorithme, permet de l'étendre à d'autres problèmes de placement de contrôleurs en prenant en compte d'autres métriques. Le nombre d'objectifs peut facilement être étendu éventuellement en mettant en œuvre la version NSGA III. De plus, la stratégie dynamique que nous avons proposée peut être utilisée pour résoudre des scénarios de placement *en ligne*.

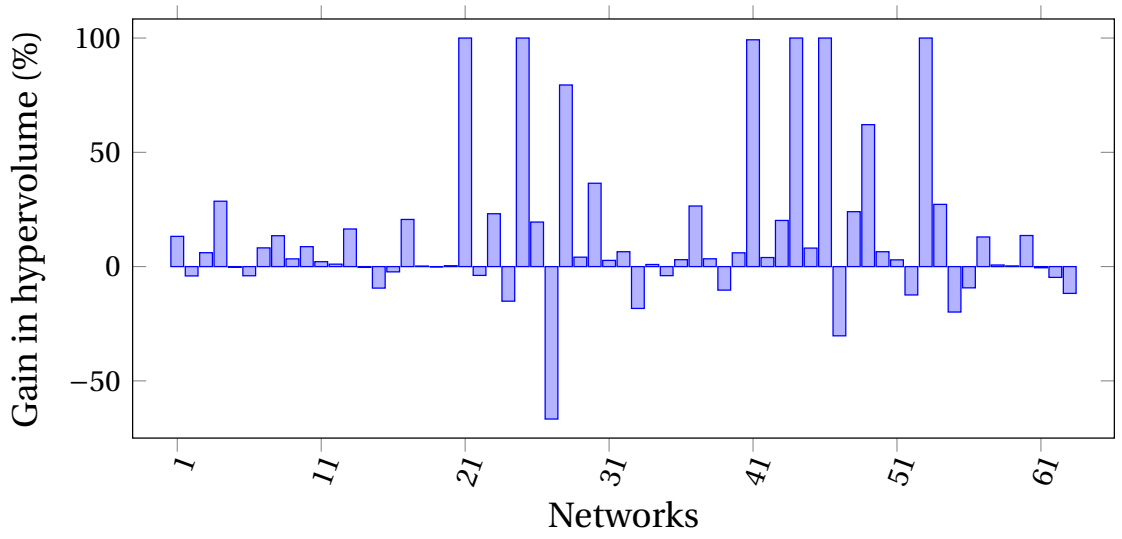


Figure 4.48 – gain en hypervolume comparé à la version mutation seule

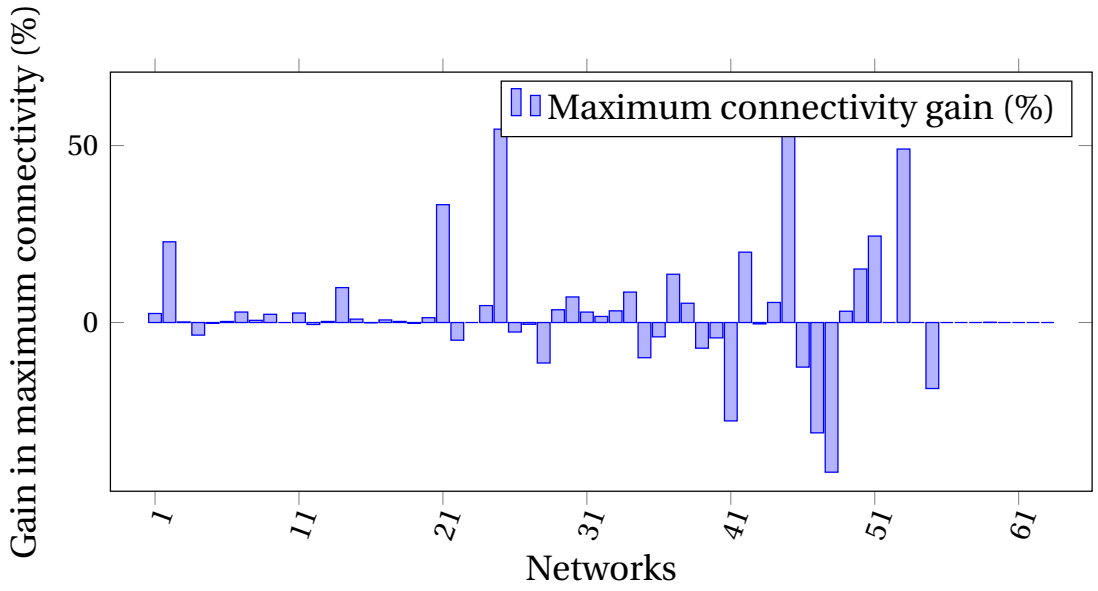


Figure 4.49 – Gain de connectivité maximal comparé à la version mutation seule

Enfin, les versions parallélisables des algorithmes évolutionnaires [SHARMA et COLLET, 2010] promettent également des gains de temps de calcul très élevés.

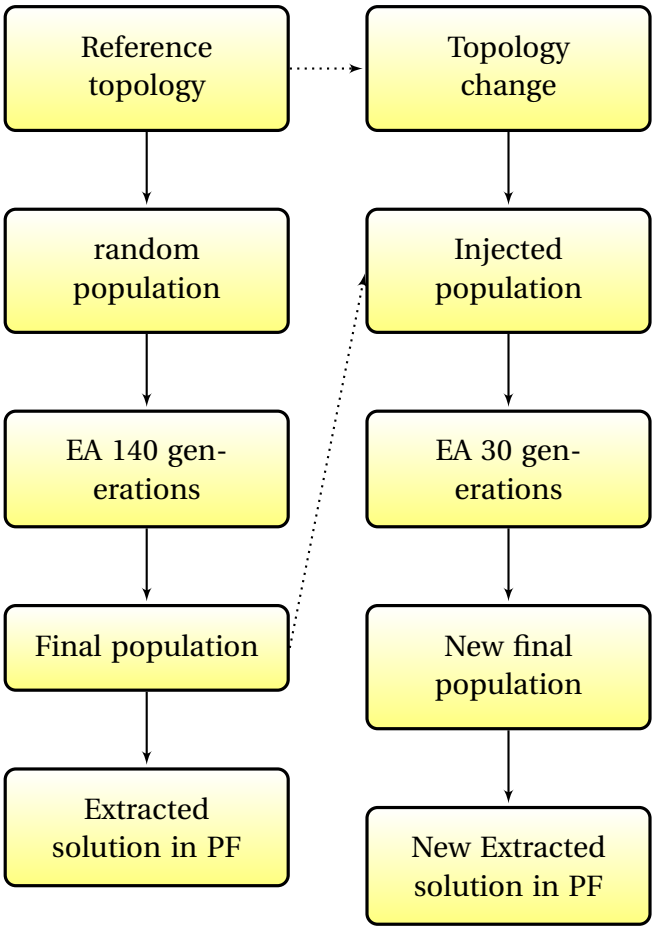


Figure 4.50 – Accélérer l'évolution

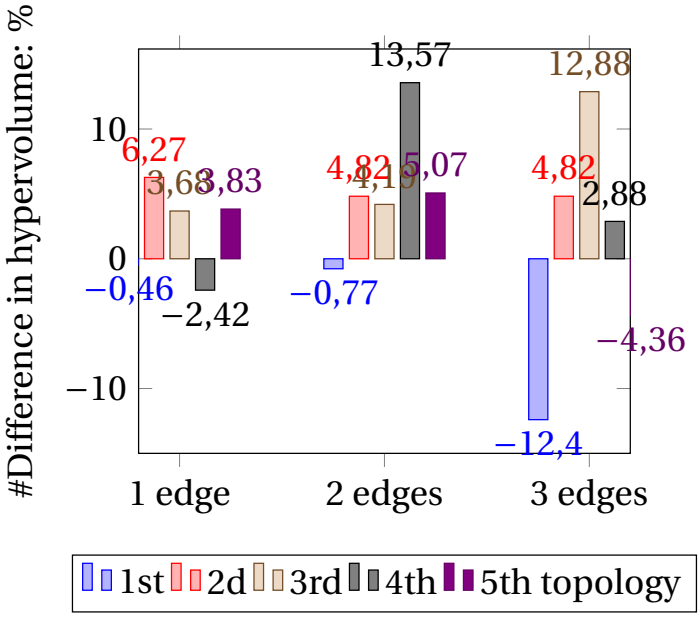


Figure 4.51 – Gain en hypervolume pour 5 topologies avec de 1 à 3 arcs enlevé

4.5 Placement de chaînes de VNFs avec un algorithme évolutionnaire

Les travaux qui sont présentés ici de manière synthétique ont été réalisés dans le cadre d'une collaboration autour des thématiques de la thèse dans le contexte d'un postdoc porté par M. Quang Pham tran anh. M. Quang Pham Tran Anh a apporté ses idées et ses compétences pour développer la thématique du placement initialement orientée sur le placement des contrôleurs aux sujets du routage et du placement de chaînes de VNF en s'appuyant sur les AEMO.

L'idée était dans la continuité des travaux de thèse autour du placement des contrôleurs d'évaluer si les AEMO pouvaient fournir une métaheuristique capable de résoudre une variété de problèmes de placements et d'allocation de ressources plus larges.

Le premier sujet que nous avons décidé d'aborder est le problème du routage multi-contraint et multi-objectifs.

Ce problème peut-être formulé comme un problème de placement de chemins qui consomment la ressource de bande passante dans le réseau.

Puis, en second lieu, nous avons envisagé le problème du placement de chaînes de VNF ou de VNF-FG qui peut être vu comme une agrégation de plusieurs problèmes d'allocation de ressources.

On peut en particulier le voir comme la combinaison d'un problème de type bin-packing pour le choix des nœuds de l'infrastructure qui supportent les VNFs et comme un problème de placement de chemins pour le choix des chemins entre les VNFs.

La manière dont nous avons abordé le problème du placement de VNF-FGs est en fait très proche de la manière dont nous avons abordé le problème du routage multi-contraint et multi-objectifs.

Le problème du routage avec contraintes multiples peut donc être vu ici comme un préalable.

4.5.1 Placement de chemins avec contraintes multiples et multi-objectifs

Le problème du routage multi-contraint est un problème très connu et exploré. Il est décrit notamment par [VAN MIEGHEM et KUIPERS, 2004] qui proposent SAMCRA, une heuristique de résolution très performante.

Certains des concepts de SAMCRA sont décrits succinctement dans l'état de l'art en 2.3.2, car ils peuvent être utiles à la compréhension de la démarche.

Nous avons voulu regarder si un AEMO pouvait traiter de manière efficace le problème avec des temps de calculs compétitifs.

Description du problème et modèle

Le problème posé est en résumé un problème de routage avec qualité de service pour les flux routés. Il s'agit de trouver un chemin optimal dans un réseau modélisé sous forme de graphe, entre un point origine et un point destination.

Différentes métriques sont prises en compte qui ont généralement une propriété d'additivité. C'est-à-dire que pour une métrique donnée, la métrique du chemin vaut la somme des métriques sur les différents arcs.

On cherche ensuite généralement à minimiser chacune des métriques et à être conforme à certaines contraintes. Différentes variantes du problème existent suivant que

l'on cherche à respecter les contraintes, optimiser les métriques ou réaliser les deux simultanément.

Le problème posé peut être vu comme une extension du problème de plus court chemin étendu à différentes métriques additives, qui correspondent à différents objectifs que l'on veut optimiser.

On peut citer comme exemples de métriques, la latence, le taux de perte (dans ce cas il s'agit d'une métrique multiplicative, mais elle peut être rendue additive via une transformation logarithmique), une métrique de coût attachée aux liens, la gigue ...

Certaines métriques ne peuvent pas être rendues additives comme la bande passante qui est comprise entre des valeurs min et max qui correspondent aux capacités des liens. Mais une telle contrainte peut être traitée en élaguant le graphe des arcs qui ne respectent pas la bande passante minimum demandée. Si on veut optimiser un objectif global de bande passante, il faut cependant s'y prendre autrement.

On ne s'intéresse donc ici en résumé qu'à des métriques additives.

CROWCROFT [1996] indique que le problème que nous évoquons est NP- difficile. Cependant, dans la pratique, il l'est rarement et les exemples de problèmes pour lesquels la résolution se heurte à la NP-complexité sont généralement créés artificiellement (VAN MIEGHEM et KUIPERS [2004]). Cette caractéristique justifie la construction d'une heuristique de résolution pratiquement exacte.

Modèle Nous présentons succinctement la modélisation du problème. Le réseau est modélisé comme un graphe orienté G = (V, E), où V représente l'ensemble des nœuds et E est l'ensemble des liens orientés. Une variable binaire Φ_e est introduite pour indiquer l'utilisation d'un lien e pour le chemin. Les mesures de QoS sont représentées par un vecteur $\{W_{i,e}, i = 1, ..., M\}$, où M est le nombre de mesures de QoS. La mesure de QoS de bout-en-bout i d'un chemin est bornée par une borne supérieure W_i. Comme nous ne considérons que les métriques additives nous avons:

$$\sum_e \Phi_e W_{i,e} \leq W_i, \forall i. \tag{4.29}$$

L'objectif est de minimiser l'ensemble des mesures de QoS, c'est-à-dire chercher:

$$\min \sum_e \Phi_e W_{i,e}, \forall i = 1 \ldots M \tag{4.30}$$

Le coût d'un chemin $l(\mathbf{P})$ est un vecteur de dimension M défini comme suit:

$$l(\mathbf{P}) = [l_1(\mathbf{P}), l_2(\mathbf{P}), \ldots, l_M(\mathbf{P})] \tag{4.31}$$

Où l'on a du fait des métriques additives: $l_i(\mathbf{P}) = \sum_e \Phi_e w_{i,e}$. L'objectif est de trouver l'ensemble des solutions non dominées c'est-à-dire Pareto optimales.

Le problème à résoudre

Nous cherchons à résoudre ni plus ni moins que le problème traité avec l'algorithme SAMCRA en utilisant un algorithme évolutionnaire. C'est-à-dire qu'on recherche un chemin optimal dans un graphe G(V, E), par rapport à des métriques additives attachées aux liens. À chaque arc du graphe est attaché un vecteur w de M mesures de QoS.

Les mesures de QoS peuvent être classées en trois catégories différentes : additive, multiplicative et min-max QoS. Les mesures multiplicatives peuvent être converties en mesures additives en prenant le logarithme. Pour les mesures de QoS min-max, la

longueur d'un chemin est le minimum ou le maximum des mesures de QoS des liens formant ce chemin. Un filtre peut être introduit pour élaguer le réseau en supprimant les liens non adaptés. Rappelons, que comme pour SAMCRA nous examinons les cas où la longueur d'un chemin est composée de mesures additives de QoS. En raison de cette additivité, nous avons:

$$\sum_e \Phi_e \mathrm{W}_{i,e} \leq \mathrm{W}_i, \forall i. \tag{4.32}$$

Les objectifs à minimiser sont toutes les mesures de QoS $\min \sum_e \Phi_e \mathrm{W}_{i,e}$, for $i = 1 \ldots k$.

Comme nous l'avons déjà indiqué, le coût d'un chemin $l(\mathbf{P})$ est un vecteur de dimension M défini comme suit:

$$l(\mathbf{P}) = [l_1(\mathbf{P}), l_2(\mathbf{P}), \ldots, l_{\mathrm{M}}(\mathbf{P})], \tag{4.33}$$

où $l_i(\mathbf{P}) = \sum_e \Phi_e \mathrm{W}_{i,e}$. L'objectif est de déterminer l'ensemble des solutions non dominées c'est-à-dire le front de Pareto.

Implémentation

Modèle pour un individu Comme nous l'avons indiqué nous avons adopté ici un modèle où un individu est un mot binaire qui représente la liste de tous les arcs du graphe. Un arc est sélectionné si le bit correspondant est passé à 1 . Il s'agit d'une modélisation classique qui correspond à ce qui est fait généralement avec les algorithmes génétiques. Le génome est donc un mot binaire.

Lors de l'initialisation de la population de manière aléatoire on voit que les solutions tirées peuvent être viables ou non viables. En fait un tirage aléatoire ne permet pas de garantir que l'individu sélectionné constituera un chemin viable, il peut être discontinu c'est-à-dire tout simplement ne pas être un chemin.

La stratégie adoptée ici pour résoudre cette difficulté a été d'introduire le concept d'opérateur de réparation.

Le concept d'opérateur de réparation L'idée est d'introduire dans la logique de l'algorithme évolutionnaire un module de réparation (réparateur) qui va transformer les individus non viables en individus viables. L'idée du réparateur est aussi ici de favoriser les solutions non dominées au sens où elles sont décrites dans le paragraphe de l'état de l'art sur SAMCRA en 2.3.2 et dans le paragraphe sur les AEMO.

Le réparateur par conséquent ici n'est pas seulement un réparateur mais aussi un module d'optimisation local dont on peut espérer qu'il va accélérer la vitesse de convergence de l'algorithme en réduisant l'espace de recherche.

Le réparateur s'appuie sur deux principes assez intuitifs qui permettent d'éliminer des arcs pour des chemins dont on sait qu'ils sont dominés. Ils sont démontrés dans [ANH QUANG et collab., 2018]. Il s'agit d'une généralisation de l'idée qu'un plus court chemin est nécessairement composé de segments qui sont des plus courts chemins.

Théorème 1 *Un chemin non dominé est un chemin simple et ne contient pas de boucles.*

Théorème 2 *On considère l'ensemble des plus courts chemins pour les différentes métriques entre une source et une destination. Pour chacune des métriques il existe une borne supérieure parmi ces différents chemins. Si un arc donné sur un chemin donné entre la source et un nœud intermédiaire vers la destination majore cette borne supérieure pour une des métriques , un chemin qui contiendrait cet arc ne peut pas être non dominé.*

Description de la logique du réparateur La logique du réparateur est présentée dans le diagramme 11. En entrée du réparateur on a un chemin non valide entre la source n et la destination d, du fait du processus d'initialisation aléatoire de la population ou du fait du mécanisme de mutation qui peuvent fabriquer des individus non valides. En sortie on dispose d'un chemin valide entre n et d.

Entre les lignes 3 à 4, tous les arcs du chemin sont stockées dans l'ensemble $\tilde{\mathscr{E}}$. Ensuite, le réparateur trouve le chemin le plus court entre n et d pour chaque mesure de QoS en utilisant l'algorithme de Dijkstra entre les lignes 5 et 6. Le résultat de cette étape sera exploité pour éliminer les chemins dominés dans les étapes ultérieures. Entre les lignes 8 et 26 on trouve la boucle principale du processus dans laquelle tous les nœuds sur le chemin entre la source et la destination sont déterminés.

Pour chaque arc sortant du nœud source n, le réparateur vérifie si la bande passante de l'arc, b_e, est suffisante (i.e. $b_e \geq \mathrm{B}_{req}$). Sinon, cet arc ne sera pas pris en compte (lignes 12 et 13).

Le processus vérifie ensuite si la destination intermédiaire n' a déjà été visitée (lignes 15 et 16) puisque le chemin Pareto-optimal ne peut pas contenir de boucles comme indiqué dans le théorème 1.

Le processus de la ligne 17 à la ligne 21 est en charge de vérifier si l'arc courant pourrait appartenir à un chemin non dominé suivant le théorème 2. S'il peut exister un chemin non dominé vers la destination à partir de n', l'arc e sera ajouté dans l'ensemble des arcs candidats $\mathscr{E}$. Enfin sinon, le réparateur sélectionne arbitrairement un lien parmi les entrées courantes de $\tilde{\mathscr{E}}$ et $\mathscr{E}^*$ ce qui permet de conserver un caractère aléatoire au processus. La complexité du réparateur n'est pas négligeable et vaut:

$$\mathrm{O}(|\mathrm{V}|^3 \cdot (\mathrm{M} + log|\mathrm{V}|)).$$

Opérateur de recombinaison L'opérateur de recombinaison mis en œuvre dans cette approche fonctionne en recherchant dans deux parents potentiels des nœuds communs. Un nœud commun est appelé nœud de recombinaison. L'idée est que s'il existe un nœud de recombinaison on peut créer deux enfants en assemblant les segments qui se trouvent avant et après le nœud de recombinaison sur chaque parent. Une condition cependant est que le nœud de recombinaison soit isolé c'est-à-dire que l'intersection des segments définis par ce point isolé dans les deux parents soit vide c'est-à-dire qu'elle ne comprenne pas d'autres nœuds communs. Ceci est démontré en [ANH QUANG et collab., 2018] et est illustré dans le diagramme 4.52. Dans ce diagramme, il y a deux parents qui correspondent aux chemins SBCAD et SACD, C est un point de recombinaison non isolé du fait de l'existence de A qui lui est un point de recombinaison isolé. La progéniture définie par SACAD est dominé par SAD qui sera le chemin non dominé retenu.

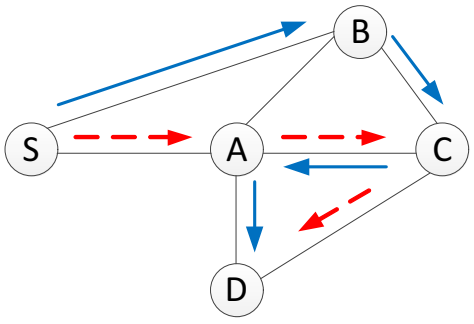


Figure 4.52 – Exemple de recombinaison de chemins

Algorithm 11: Repairer

Input: A unfeasible path from source S to destination D
Output: A feasible path from source S to destination D
foreach *active bit of the input* **do**
    $\tilde{\mathscr{E}} \leftarrow e$;
foreach *QoS measure i* **do**
    Run Dijkstra algorithm to find shortest path in terms of i from d to all other nodes $\rightarrow \tilde{c}_i^{n,d}$;
$n \leftarrow s$;
while $n \neq d$ **do**
    $\mathrm{V}^* \leftarrow n$;
    $\mathscr{E}^* \leftarrow \emptyset$;
    foreach *e originated from n* **do**
        if $b_e < \mathrm{B}_{req}$ **then**
            continue;
        $n' \leftarrow \mathrm{D}est(e)$;
        if $n' \in \mathrm{V}^*$ **then**
            continue;
        foreach *QoS measure i* **do**
            $u_i^{n,d} \leftarrow \max_{j \neq i} \tilde{c}_{j,i}^{n,d}$;
            if $\tilde{c}_{i,i}^{n',d} + c_i^{n,n'} \leq u_i^{n,d}$ **then**
                $\mathscr{E}^* \leftarrow e$;
                break;
    if $\mathscr{E}^* \cap \tilde{\mathscr{E}} \neq \emptyset$ **then**
        Select an edge e among entries in $\mathscr{E}^* \cap \tilde{\mathscr{E}}$ randomly;
    else
        Select an edge e among entries in $\mathscr{E}^*$ randomly;
    $n \leftarrow \mathrm{D}est(e)$;

Description de la logique de l'opérateur de recombinaison L'opérateur de recombinaison commence par rechercher les nœuds de recombinaison dans les deux parents. Il regarde ensuite s'il est possible de créer une progéniture non dominée à partir des deux parents. Les différents enfants sont générés et choisis ensuite aléatoirement. Si il n'existe pas de progéniture possible, un des deux parents est choisi au hasard comme progéniture.

La complexité de l'opérateur de recombinaison vaut:

$$\mathrm{O}(|\mathrm{V}| \cdot (|\mathrm{V}| \cdot (|\mathrm{V}| \cdot log|\mathrm{V}|) + \mathrm{M}).$$

Structure de l'algorithme La principale particularité est le module réparateur qui intervient après l'initialisation et après l'étape de mutation. Le diagramme 4.53 illustre la structure de l'algorithme qui est la structure classique d'un algorithme évolutionnaire dans laquelle on introduit le réparateur pour rendre viable les individus non viables.

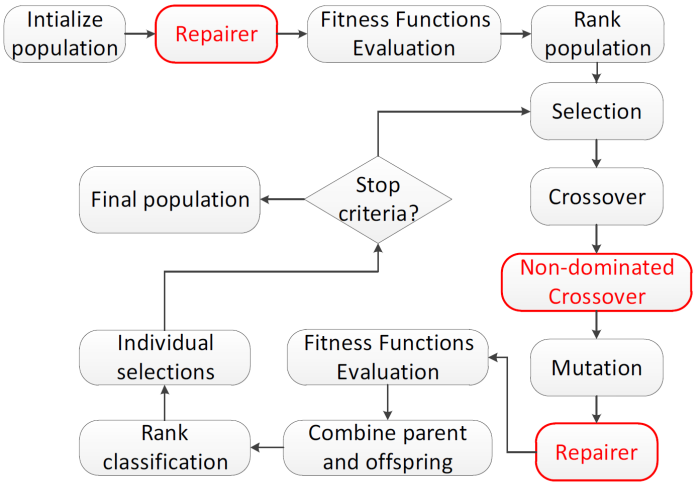


Figure 4.53 – Algorithme génétique proposé

Résultats obtenus

L'analyse des performances de l'algorithme a été effectuée à partir de 5 réseaux de caractéristiques de tailles variables, provenant de la base de données zoo-topology. Les caractéristiques de ces réseaux sont résumées dans la table 4.6.

Nom	Nb nœuds	Nb arcs	Shortest path distance
Kdl (Kentucky Data Link)	754	1788	39
Colt (Colt telecom)	153	354	18
GtsCe (GTS Central Europe)	149	386	27
UsCarrier (US Carrier)	158	378	26
Deltacom (ITC Deltacom)	113	322	18

Table 4.6 – Paramètres des réseaux évalués

D'autre part, la table 4.7 synthétise le paramétrage de l'algorithme pour exécuter les tests.

Paramêtre	Valeur
Population	20
Générations	120
QoS metrics	Loss rate and Latency
Loss rate	0.01% – 0.1% [RODRIGUEZ-COLINA et collab., 2014]
Nombre d'itérations par scénario	30
Intervalle de confiance	95%
Environnement de simulation	Intel Core i5-6300U, 8GB RAM
Framework algorithmique	jMetal [NEBRO et collab., 2015]

Table 4.7 – Paramètres de simulation

Différentes options de mise en œuvre de l'AE ont été testées.

- Une option intégrant l'opérateur de mutation et l'opérateur de recombinaison qui prend en compte la dominance, en privilégiant les progénitures non-dominées: NDC-M (Non Dominated Cross over and mutation),
- une option avec l'opérateur de recombinaison qui prend en compte la dominance sans mutations (NDC),
- Une option avec l'opérateur de mutation seul (M),
- Une option avec l'opérateur de mutation et un opérateur de recombinaison qui utilise des points arbitraires de recombinaison (SP-M) et le réparateur .

L'avantage de l'option NDC est qu'elle ne nécessite pas la mise en oeuvre du réparateur à chaque étape (hormis à l'initialisation) puisque les solutions proposées restent valides, ce qui permet de réduire la complexité.

Dans l'analyse des résultats, les performances du AEMO on été comparées avec l'algorithme SAMCRA adapté pour fournir un front de Pareto dénommé (SAMCRA-O-MO) en intégrant l'option "anticipation" (*look ahead*) qui permet d'accélérer les traitements. Les différentes versions de l'algorithme sont donc évaluées par rapport au front de Pareto fourni par SAMCRA. On définit pour cela un hypervolume normalisé (NHV) qui est le rapport de l'hypervolume de l'AEMO à l'hypervolume fourni par SAMCRA. Meilleur est ce rapport, meilleures sont les performances de l'AE pour la résolution du front de Pareto.

Hypervolume normalisé Les deux figures en 4.54 présentent les résultats obtenus pour les 5 réseaux dans deux configurations: une population de 20 individus et 120 générations, et une population de 10 individus et 60 générations. Il y a trois métriques additives qui sont testées.

On remarque tout d'abord que l'on est pour toutes les configurations à mieux que 80% du front de Pareto obtenus avec SAMCRA-LO-MO bien que l'on utilise une très petite population et un nombre relativement raisonnable de générations.

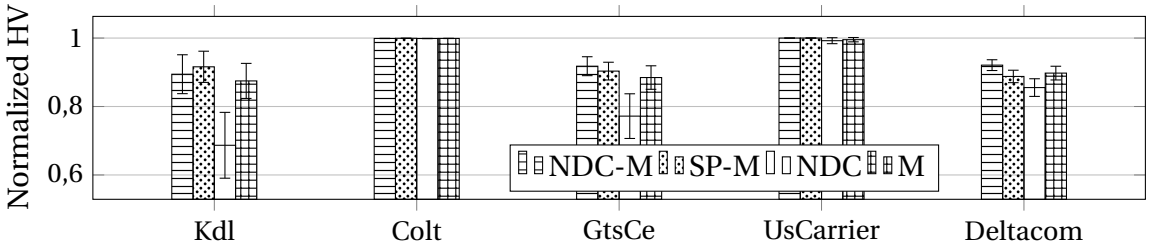
On remarque ensuite que la version NDC-M donne les meilleurs résultats mais ces résultats sont globalement très proches des variantes SP-M et M.

On remarque enfin une différence pour le réseau Kdl qui est le plus gros réseau et où la variante NDC donne de moins bons résultats avec ce qui semble être une convergence vers un minimum local.

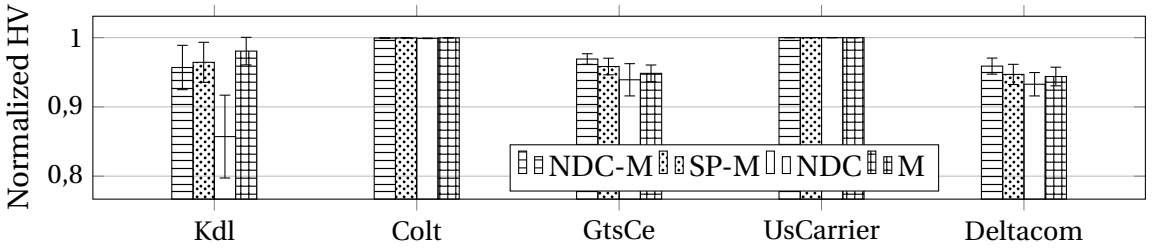
Temps de calcul Le diagramme 4.55 compare les performances obtenues en terme de temps de calcul. Le résultat essentiel à retenir est que la variante NDC est plus performante parce qu'elle ne fait pas appel au réparateur qui est pénalisant dans la mesure où il peut intervenir en cas de mutations à chaque génération.

Le temps de calcul est aussi comparé à SAMCRA pour les variantes NDC et SP-M dans le diagramme 4.56 où l'on voit que l'algorithme évolutionnaire peut même être plus performant que SAMCRA en terme de temps de calcul pour un gros réseau où pour SAMCRA il commence à devenir significatif.

Front de Pareto Les front de Pareto obtenus avec NDC-M et SAMCRA pour les deux métriques que sont la latence et le taux de perte sont comparés dans le diagramme 4.57 pour les réseaux Kdl, Colt et GTS. On peut remarquer que l'algorithme évolutionnaire ne retrouve pas tous les points du front de Pareto dans le cas du réseau Kdl ce qui explique la valeur moins élevée de l'hypervolume normalisé obtenu pour ce réseau.

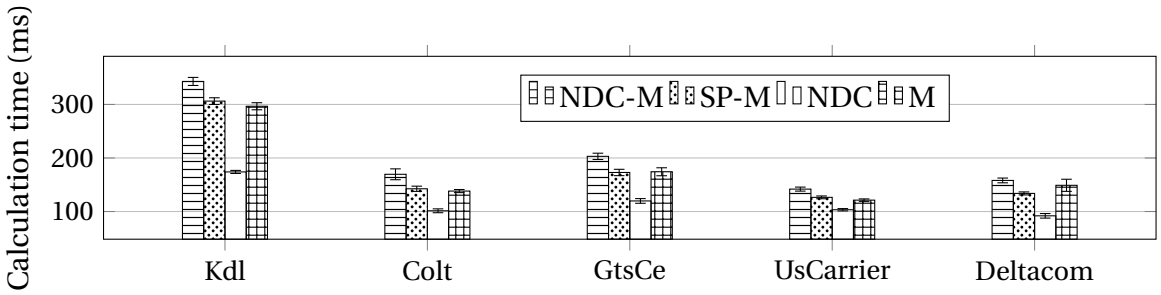


(a) Hypervolume normalisé avec 10 individus et 60 générations

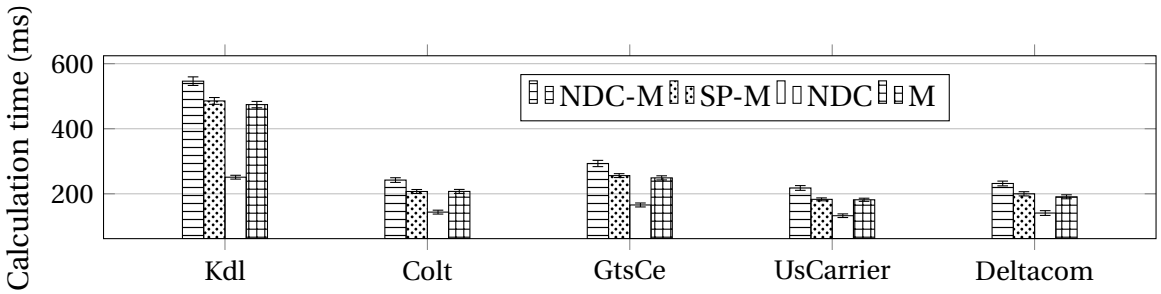


(b) Hypervolume normalisé avec 20 individus et 120 générations

Figure 4.54 – Hypervolume normalisé, 3 métriques de QoS



(a) Temps de calcul avec 10 individus et 60 générations



(b) Temps de calcul avec 20 individus et 120 générations

Figure 4.55 – Temps de calcul, 3 métriques de QoS

Comparaison de différents algorithmes évolutionnaires Les principes de cette approche ont été insérés dans la structure de différents AEMO à des fins de comparaisons. L'intérêt des AE est en effet leur structure flexible.

Les algorithmes comparés sont NSGA-II , NSGA-III qui est une amélioration de NSGA-II visant en particulier à de meilleures performances quand les objectifs sont nombeux [CIRO et collab., 2016] et aussi SPEA-II [ZITZLER et collab., 2001] qui est historiquement l'algorithme concurrent de NSGA-II. Le réseau testé est Kdl, c'est-à-dire le plus gros réseau. Nous présentons les résultats dans les diagrammes 4.59 pour l'hypervolume normalisé et 4.58 pour le temps de calcul.

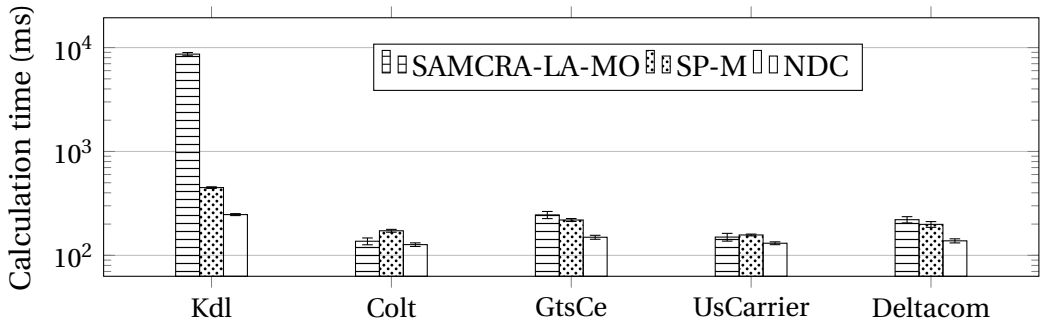


Figure 4.56 – Temps de calcul de SAMCRA-LA-MO vs GAs (20 individus et 120 générations), 2 métriques de QoS

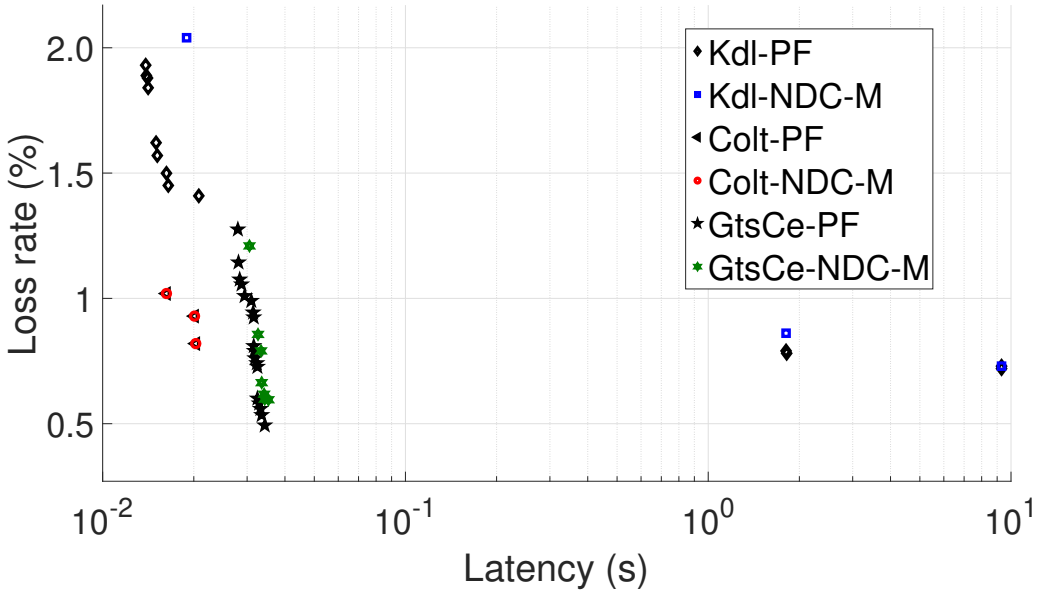


Figure 4.57 – Taux de perte et latence

Les résultats montrent que NSGA-II est moins bon du point de vue de l'approche du front de Pareto par rapport à NSGA-III et SPEA-II, tandis que le temps de calcul est en faveur de NSGA-II. SPEA-II est étonnamment bon du point de vue de l'approche du front de Pareto, alors qu'il est théoriquement surpassé par NSGA-II.

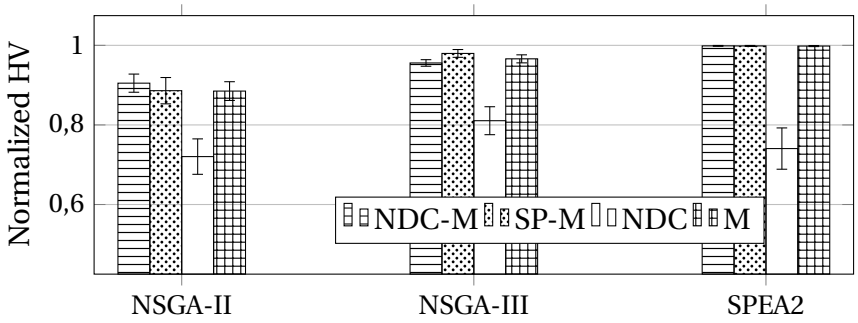


Figure 4.58 – HV normalisé (10 individus et 60 generations), 2 métriques de QoS pour différentes MOEA

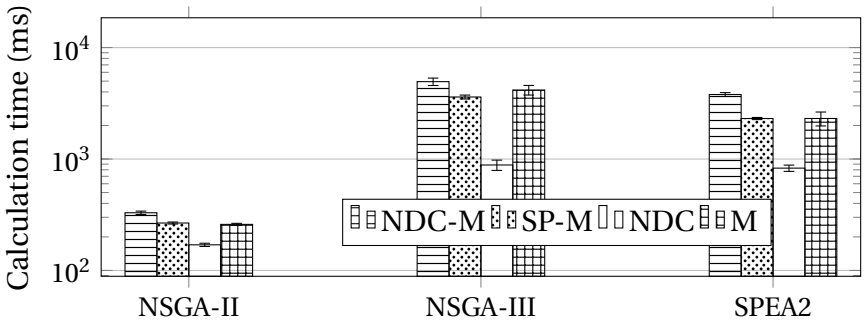


Figure 4.59 – Temps de calcul (10 individus et 60 générations), 2 métriques de QoS sur différents AE

Parralélisme La librairie JMetal [NEBRO et collab., 2015] que nous avons utilisée permet de paralléliser l'étape d'évaluation des individus dans la population sur différents cœurs. Dans l' approche proposée, l'étape la plus pénalisante en terme de temps de calcul est l'application du réparateur sur chaque individu. Nous avons donc parallélisé le processus du réparateur.

Des simulations ont été effectuées pour vérifier les avantages du NSGA-II avec le réparateur parallélisé, noté p*NSGA-II, et de la parallélisation par défaut de NSGA-II dans jMetal, noté pNSGA-II. Le réseau testé est de nouveau le réseau Kdl, qui est celui de plus grande taille.

Le nombre de générations et la taille de la population sont respectivement de 60 et 10. Les résultats sont illustrés dans la Figure 4.60. Lorsque seul le mécanisme NDC est utilisé, la différence de temps de calcul n'est pas significative, toujours parce que le processus du réparateur est exécuté une seule fois, lorsque la population initiale est générée. Entretemps, il y a des différences notables dans les configurations où le processus de réparation doit être exécuté à chaque génération.

En général, p*NSGA-II est capable de réduire la durée d'exécution d'environ 30%, grâce à la parallélisation du réparateur.

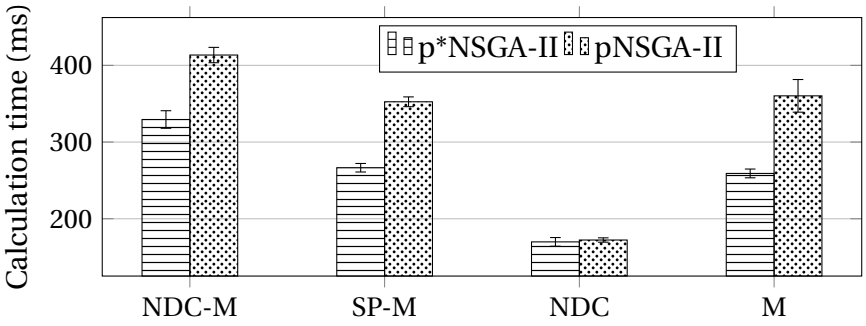


Figure 4.60 – Temps de calcul de p*NSGA-II and pNSGA-II

Conclusion sur le routage multi-contraint & multi-objectifs

Ce travail a montré qu'un algorithme évolutionnaire pouvait être aussi performant qu'une heuristique spécialisée qui est ce qui se fait de mieux sur le sujet. Il faut cependant nuancer le propos, cette performance a été obtenue en cassant, c'est-à-dire en spécialisant l'AEMO en ajoutant un opérateur de réparation qui intègre un mécanisme d'optimisation

locale inspiré de SAMCRA. Cette spécialisation est contradictoire par rapport à l'objectif de généricité que nous nous sommes fixés.

On remarque aussi que l'algorithme n'est pas réellement meilleur du point de vue du front de Pareto lorsqu' est introduit l'opérateur de recombinaison par rapport à la version mutation seule. Cela montre que celui-ci n'est pas réellement efficace pour relayer les bonnes propriétés du génome des parents.

4.5.2 Placement de chaînes de VNF

Nous avons abordé dans la continuité du travail précédent le problème du placement de chaînes de VNF de manière similaire au problème du routage multi-contraint et multi-objectif dans le sens où nous cherchons à optimiser les métriques de QoS attachées au service réseau représenté par le VNF-FG.

Le problème posé ici consiste donc à rechercher le placement optimal de la chaîne de fonctions du service réseau qui va optimiser les différentes métriques du service. D'autre part, nous intégrons dans le modèle une fonction de coûts, somme des coûts relatifs attachés aux nœuds et aux liens, que nous appelons coût du déploiement.

Cette fonction de coûts si elle est calibrée permet de disposer d'une métrique pour optimiser des paramètres globaux comme la consommation énergétique ou la bande passante résiduelle.

Elle peut permettre aussi de mieux répartir les nœuds et liens virtuels du VNF-FG dans le graphe de l'infrastructure en fonction des capacités disponibles.

Les métriques sont des métriques qui caractérisent des consommations de ressources soit au niveau des nœuds de l'infrastructure (consommation CPU, mémoire et stockage de la VNF), soit au niveau des liens (bande passante) et des métriques qui caractérisent aussi le chemin réseau que vont emprunter les flux pour bénéficier du service réseau.

L'approche adoptée est une mise à l'échelle verticale, c'est-à-dire que pour une demande de flux de trafic fixée, on va dimensionner de manière adhoc la capacité des VNFs du VNF-FG et rechercher les nœuds et les liens dans l'infrastructure qui permettent d'accueillir ces VNFs et de réaliser cette demande.

Un VNF-FG est donc caractérisé par la topologie du graphe de transfert (FG), les ressources consommées par chaque VNF et chaque lien du VNF-FG pour répondre à la demande de trafic, et enfin par l'ordre de parcours des VNFs dans le graphe par le flot réseau.

La démarche est illustrée par le diagramme 4.61. Nous verrons que l'algorithme utilise comme sous-routine notre algorithme évolutionnaire de calcul de chemins multi-contraints désigné par *GAR* dans la suite du texte.

Modèle

L'infrastructure réseau est modélisée comme un graphe orienté $G_s = (V_s, E_s)$, où V_s désigne l'ensemble des nœuds et E_s est l'ensemble des arcs.

Un VNF-FG est représenté sous la forme d'un graphe orienté $G_v = (V_v, E_v)$, ou V_v représente l'ensemble des VNFs et E_v est l'ensemble des liens virtuels orientés reliant les VNFs dans les VNF-FGs. Nous introduisons les variables binaires $\Phi_e^{e'}$ and $\Omega_n^{n'}$ pour indiquer si un lien orienté e est utilisé par le lien virtuel e' et la VNF n' est supportée par le nœud n de l'infrastructure.

Les ressources de l'infrastructure peuvent être classées en deux types : les ressources réseau et les ressources de calcul et de stockage. Les ressources réseau comprennent les

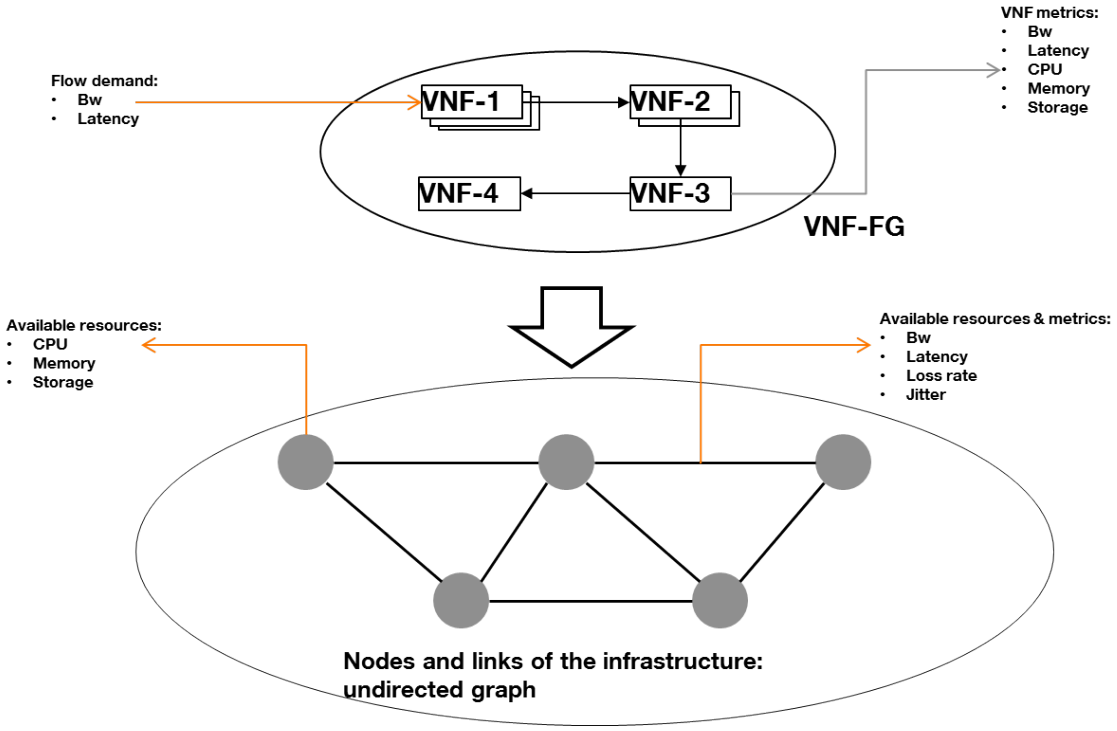


Figure 4.61 – Placement d'une chaîne de VNFs dans une infrastructure

Notation	Description
w_i^e	i^{th} measure of link e
$\Gamma_i^{e'}$	Requirement of i^{th} measure of virtual link e'
$\Psi_i^{n'}$	Requirement of i^{th} resource of VNF n'
v_i^n	Cost of resource i^{th} of substrate node n
z_i^e	Cost of resource i^{th} of substrate link e
r_k^n	Resource of k^{th} measure of substrate node n

Table 4.8 – Notations pour les métriques de QoS, les ressources, et le coût de déploiement

caractéristiques des liens reliant les nœuds de l'infrastructure, tandis que les ressources de calcul et de stockage sont la puissance de calcul (CPU), la mémoire vive (RAM) et la quantité de stockage disponible. Toutes les notations utilisées dans cette formulation du problème sont présentées dans le tableau 4.8.

Nous considérons en premier lieu les contraintes liées aux ressources du réseau. Les métriques du lien e sont présentées par un vecteur $\mathbf{w}^e = \{w_i^e, i = 1, ..., \mathrm{M}\}$, où M est le nombre de métriques. Les indices e et (n, m) (où n et m sont respectivement la source et la destination du lien e) sont utilisés dans la suite du texte.

Comme dans le paragraphe précédent sur le routage avec contraintes, nous nous concentrons ici sur des métriques additives. Ces mesures peuvent être des mesures de la qualité de service (par exemple la latence) ou les coûts de l'utilisation des liens. Ici, nous considérons la latence et le taux de perte. Chaque lien virtuel a ses mesures spécifiques, notées $\Gamma_i^{e'}$.

Les contraintes de QoS de chaque lien virtuel peuvent être formulées comme suit:

$$\sum_e \Phi_e^{e'} w_i^e \leq \Gamma_i^{e'}, \forall i, e'. \tag{4.34}$$

Ensuite, les ressources de calcul et de stockage sont prises en compte. Chaque nœud

n de l'infrastructure a des types de ressources k désignés par $1,...,k$. La quantité de ressource k du nœud n est indiquée par r_k^n. Nous considérons trois types de ressources , la ressource CPU, la RAM et la mémoire de stockage. La quantité de ressource k demandée par la VNF n' est notée par $\Psi_k^{n'}$.

Les contraintes des ressources de calcul et de stockage sont modélisées de la manière suivante:

$$\sum_{n'} \Omega_n^{n'} \Psi_k^{n'} \leq r_k^n, \forall n, k \tag{4.35}$$

De plus chaque VNF est portée par un seul nœud de l'infrastructure réseau. D'où la contrainte suivante:

$$\sum_{n} \Omega_n^{n'} \leq 1, \forall n' \tag{4.36}$$

On désigne l'ensemble des liens entrants sur un nœud n comme $\mathbb{I}_n$ et l'ensemble des liens sortants du nœud n comme $\mathbb{O}_n$. Une connexion entre deux VNF adjacentes imposée par le modèle VNF-FG peut être modélisée en s'inspirant du problème multi-flots (*multi-commoditiy flows problem*) et de la contrainte de continuité grâce à l'équation 4.37.

$$\sum_{e \in \mathbb{O}_n} \Phi_e^{e'} - \sum_{\tilde{e} \in \mathbb{I}_n} \Phi_{\tilde{e}}^{e'} = \Omega_n^{n'} - \Omega_n^{m'}, \forall n \tag{4.37}$$

Nous cherchons à minimiser toutes les métriques comme noté dans l'équation 4.38:

$$\min \sum_{e'} \sum_{e} \Phi_e^{e'} w_i^e, i = 1 \dots k \tag{4.38}$$

D'autre part nous cherchons à minimiser un coût de déploiement qui est la somme des coûts attachés aux nœuds et aux liens. Ce coût de déploiement peut être utilisé par exemple pour répartir davantage les VNFs dans les nœuds du réseau en définissant un coût plus élevé si deux VNFs ou plus sont installées sur un nœud par rapport à un scénario ou elles sont réparties sur deux nœuds.

Pour la bande passante résiduelle, on peut de la même manière pénaliser davantage quand deux liens virtuels sont sur un même lien physique plutôt que sur deux liens distincts.

Ce coût de déploiement peut être non linéaire ce qui n'est pas une difficulté pour un AE.

$$\min \sum_{k=1}^{\mathrm{K}} \sum_{n'} \sum_{n} \Omega_n^{n'} v_k^n + \sum_{i=1}^{\mathrm{M}} \sum_{e'} \sum_{e} \Phi_e^{e'} z_i^e, \tag{4.39}$$

En résumé, le problème de placement d'une VNF-FG peut être exprimé comme suit;

$$\begin{array}{ll} \textbf{Objectives} & (4.38), (4.39) \\ \textbf{s.t.} & (4.34), (4.35), (4.36), (4.37) \end{array} \tag{4.40}$$

Le problème de flots multiples (*multicommodity flow problem*) avec un seul objectif et des contraintes linéaires est un problème NP-difficile, pour lequel on ne peut pas trouver de solution optimale en temps polynomial [QUANG et collab., 2017], et de plus la complexité augmente avec l'aspect multi-objectifs.

Approche adoptée pour l'algorithme évolutionnaire

Comme pour le routage l'approche adoptée consiste à utiliser un AE.

Dans notre modèle les individus sont des listes de bits $\Phi_e^{e'}$ et $\Omega_n^{n'}$ qui indiquent si un lien de l'infrastructure physique est utilisé pour supporter chaque lien virtuel du VNF-FG et si un nœud de l'infrastructure physique est utilisé pour supporter chacune des VNFs $\Omega_n^{n'}$ du VNF-FG.

L'initialisation A l'initialisation, des VNF-FG sont crées par l'assignation aléatoire des VNFs à des hôtes disposant des ressources adéquates en satisfaisant les contraintes de ressources.

En premier lieu, l'ensemble des nœuds de l'infrastructure qui peuvent satisfaire les demandes de ressources de VNF n', $\mathbb{S}_{n'}$, sont identifiés.

Chaque VNF est allouée au maximum à un nœud, tandis qu'un nœud peut héberger plusieurs VNFs de manière à optimiser la QoS de bout en bout.

L'hôte de n' est sélectionné parmi un ensemble $\mathbb{S}_{n'}$ de manière à minimiser le nombre d'hôtes.

Algorithme de recherche de solutions non dominées L'étape d'initialisation produit un ensemble de nœuds de l'infrastructure capables d'accueillir des VNFs du VNF-FG. Pour chaque individu, il faut ensuite identifier des chemins reliant les hôtes des VNFs. En exécutant l'algorithme `GAR` on trouve un sous-chemin, reliant les hôtes de l'infrastructure pour chaque lien virtuel. Le but de l'algorithme 12 est de créer un ensemble de chemins de bout en bout pour un VNF-FG. Il utilise pour cela les théorèmes 3 et 2.

Théorème 3 *Avec des métriques additives, un chemin non dominé traversant des nœuds fixés est constitué de sous-chemins non dominé reliant ces nœuds.*

Ce principe permet d'affirmer que le chemin passant par différentes VNF sera nécessairement non dominé dans la mesure où les sous-chemins correspondants aux liens virtuels du VNF-FG sont non dominés.

En entrée de l'algorithme on a les nœuds de l'infrastructure choisis pour les nœuds virtuels du VNF-FG $\Omega_n^{n'}$.

L'algorithme commence par calculer l'ensemble des sous-chemins non dominés de chaque lien virtuel. Pour chaque lien virtuel $n^{'}m^{'}$, l'algorithme connaît la source S et la destination D dans $\Omega_{n_i}^{n'}$ (ligne 6). Il recherche ensuite dans la base de données des sous-chemins explorés $\mathbb{Q}^{'}$. S'il existe des chemins non dominés de S à D dans la base, l'algorithme les récupère (ligne 7 à 8) et supprime les chemins violant les exigences de QoS (ligne 9). Sinon, il appelle l'algorithme `GAR` pour déterminer l'ensemble des chemins non dominés entre S et D et les ajouter à la base $\mathbb{Q}^{'}$ (ligne 10 à ligne 12).

Disposer de la mémoire de l'ensemble des sous-chemins explorés aide à réduire le temps de calcul en évitant les appels trop fréquents à l'algorithme `GAR`. Les nouveaux sous-chemins seront ajoutés à l'ensemble des chemins non dominés courants, $\mathbb{P}^*$, qui sont la combinaison des sous-chemins des liens virtuels précédents (ligne 14 à 18). Le nouveau chemin formé est ajouté dans un ensemble $\mathbb{C}$ s'il satisfait aux contraintes de bande passante (c'est-à-dire que tous les liens du chemin ont une bande passante plus grande que celle demandée).

En s'appuyant sur le théorème 3, l'algorithme sélectionne uniquement les chemins non dominés de $\mathbb{C}$ (ligne 20 à ligne 24) pour mettre à jour l'ensemble des chemins non dominés déjà déterminés. L'algorithme s'exécute jusqu'à ce que tous les liens virtuels

Algorithm 12: Non-dominated Routing Algorithm

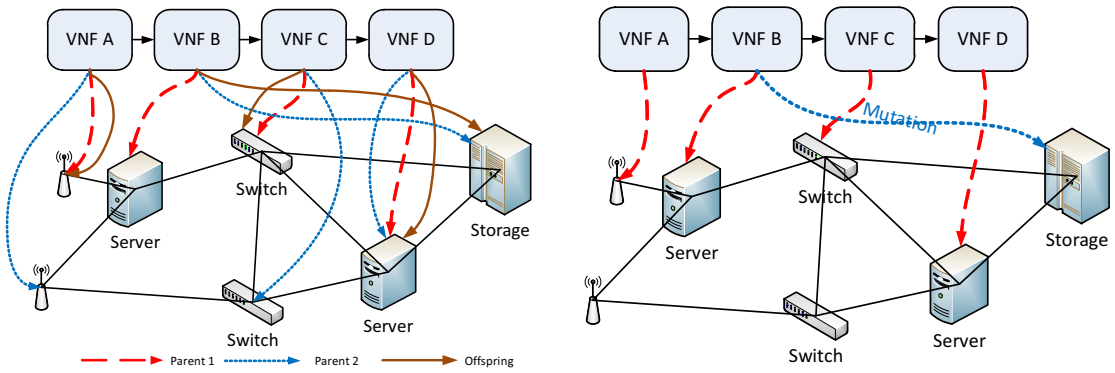
Input: A graph of VNFs and their hosts $\left(\Phi_n^{n'}\right)$, QoS requests for each virtual link $\mathbf{q}_{n'm'}$ and end-to-end $\mathbf{q}_{e2e}$
Output: non-dominated path traversing through all hosts of VNFs
End-to-end non-dominated path $\mathbb{P}^* \leftarrow \emptyset$;
Explored sub-paths $\mathbb{Q}^{'} \leftarrow \emptyset$;
foreach *Virtual link* $n'm'$ **do**
    $S \leftarrow h_{n'}$ and $D \leftarrow h_{m'}$;
    if $\mathbb{Q}^{'}.find(S,D) \neq \emptyset$ **then**
        $\mathbb{Q}^* \leftarrow \mathbb{Q}^{'}.\texttt{find}(S,D)$;
        Remove paths which violate $\mathbf{q}_{n'm'}$ in $\mathbb{Q}^*$;
    else
        $\mathbb{Q}^* \leftarrow \texttt{GAR}\left(S,D,\mathbf{q}_{n'm'}\right)$;
        $\mathbb{Q}^{'} \leftarrow \mathbb{Q}^{'} \cup \mathbb{Q}^*$;
    Set of combinations $\mathbb{C} \leftarrow \emptyset$;
    foreach $\mathscr{P} \in \mathbb{P}^*$ **do**
        foreach $\mathscr{Q} \in \mathbb{Q}^*$ **do**
            $\mathscr{P}' \leftarrow \texttt{concatenate}(\mathscr{P},\mathscr{Q})$;
            $c_{\mathscr{P}'} \leftarrow$ Compute the cost of $\mathscr{P}'$;
            if $\mathscr{P}'$ *satisfies* $\mathbf{q}_{e2e}$ **then** $\mathbb{C} \leftarrow \left(\mathscr{P}', c_{\mathscr{P}'}\right)$; ;
    $\mathbb{P}^* \leftarrow \emptyset$;
    foreach $\mathscr{P} \in \mathbb{C}$ **do**
        $n_{\mathscr{P}} = 0$;
        foreach $\mathscr{Q} \in \mathbb{C}$ **do**
            if $\mathscr{Q} \prec \mathscr{P}$ **then** $n_{\mathscr{P}} = n_{\mathscr{P}} + 1$; ;
        if $n_{\mathscr{P}} = 0$ **then** $\mathbb{P}^* \leftarrow \mathscr{P}$; ;

soient traités. Il délivre en sortie l'ensemble des chemins non dominés qui satisfont toutes les requêtes de QoS des liens virtuels ainsi que celles relatives aux chemins de bout en bout.

Individus valides et non valides Comme dans le problème du routage, les individus initialisés de manière aléatoire peuvent être non valide du fait des discontinuités. Cependant l'algorithme utilise le mécanisme présenté précédemment pour le réparateur qui permet de transformer les chemins non valides entre les VNFs en chemins valides.

Opérateur de recombinaison et de mutation L'opérateur de recombinaison introduit dans la boucle principale de l'algorithme est non standard. Il s'applique à l'emplacement des VNFs des parents en les permutant avec une probabilité θ_c.

L'opérateur de mutation change le nœud support de chaque VNF avec une probabilité θ_m. Quand l'opérateur de mutation est appliqué, l'algorithme AE de recherche de chemins non dominés est appliqué pour trouver un chemin non dominé entre chaque VNFs.



(a) Crossover operator

(b) Mutation operator

Figure 4.62 – Exemples d'opérateurs de recombinaison et de mutation

AE de placement de VNF-FG L'algorithme proposé pour le problème d'allocation des VNF-FG est présenté en 13. Il doit disposer d'informations sur les VNF-FGs, et doit donc être situé dans un orchestrateur de service NFVO. L'algorithme commence par initialiser les valeurs de $\Omega_n^{n'}$.

L'algorithme privilégie l'assignement de plusieurs VNFs aux mêmes nœuds de l'infrastructure lorsque cela permet de réduire la mesure de la QoS de bout en bout, même si cela peut nuire à l'optimisation du coût de déploiement de la VNF.

Ensuite 12 est exécuté pour déterminer un ensemble de chemins traversant tous les hôtes de l'infrastructure. Pour chaque génération, les opérateurs de variations (recombinaison et mutation) sont appliqués pour étendre l'espace de recherche. En raison des opérateurs de variation, $\Phi_e^{e'}$ doit être revue pour être cohérente avec le nouveau $\Omega_n^{n'}$.

Ceci est fait avec la routine 12. L'algorithme se poursuit jusqu'à ce que le nombre maximum de générations soit atteint.

Algorithm 13: Genetic algorithm for VNF-FG allocation

Input: The size of population N and the number of generations G, SFC
Output: Substrate hosts $\left(\Omega_n^{n'}\right)$ and links $\left(\Phi_e^{e'}\right)$ for SFC Initialize $n \leftarrow 0$, $g \leftarrow 0$;
Set of population $\mathbb{P} \rightarrow \emptyset$;
while $n <$ N **do**
    Initialize an individual p by assigning value for $\Omega_n^{n'}$ such that $\Omega_n^{n'}$ satisfies resource constraints and minimize number of substrate hosts;
    Run Algorithm 12 to identify a set of paths $\mathscr{P}$ traversing all host defined by $\Omega_n^{n'}$;
    Select a path from $\mathscr{P}$ randomly $\rightarrow \Phi_e^{e'}$;
    Increase n;
while $g <$ G **do**
    `Crossover`$\left(\Omega_n^{n'}\right) \rightarrow \Omega_n^{n'}$ `Mutation`$\left(\Omega_n^{n'}\right) \rightarrow \Omega_n^{n'}$ **if** $Mutation \neq null$ ***or*** $Crossover \neq null$ **then**
        foreach *individual* **do**
            Run Algorithm 12 to identify a set of paths $\mathscr{P}$ traversing all host defined by $\Omega_n^{n'}$;
            Select a path from $\mathscr{P}$ randomly $\rightarrow \Phi_e^{e'}$;
    Increase g;
return $\Omega_n^{n'}$ and $\Phi_e^{e'}$

Résultats

Borne supérieure de l'hypervolume L'évaluation de la qualité des solutions en utilisant l'hypervolume nécessite une référence. Les solutions produites peuvent être subdivisées entre le choix des nœuds qui supportent les VNFs et le choix des chemins qui relient ces VNFs.

Pour cela nous introduisons une notion de borne supérieure de l'hypervolume qui permettra d'évaluer les solutions par rapport à cette borne.

Notons $\mathtt{S}^*$, $\widehat{\mathtt{S}}$, et $\tilde{\mathtt{S}}^*$ l'ensemble des solutions d'allocations de VNF-FGs, solutions du problème de routage, et en solution de choix de nœuds supports des VNFs respectivement. Une solution VNF-FGs d'allocation i, $\mathtt{S}_i^* \in \mathtt{S}^*$ est constituée par deux sous solutions $\widehat{\mathtt{S}}_i^*$ et $\tilde{\mathtt{S}}_i^*$.

Théorème 4 *Il n'existe pas de solution d'allocation de VNF-FG qui a des sous-solutions dominant toutes les solutions du problème de routage et toutes les solutions du problème de sélection des nœuds dans l'infrastructure.*

Notons $\mathtt{S}^c$ comme la combinaison des solutions du problème de routage et de choix des nœuds supportant les VNFs dans l'infrastructure. L'hypervolume d'une solution $\mathtt{S}$ est notée $\mathtt{HV}(\mathtt{S})$.

Théorème 5 $HV(S^c) \geq HV(S^*)$

Ces théorèmes sont démontrés dans la publication de [PHAM TRAN ANH QUANG JEAN-MICHEL SANNER, 2018].

Du fait du théorème 5, une borne supérieure de l'hypervolume est: $\mathtt{HV}(\mathtt{S}^c)$.

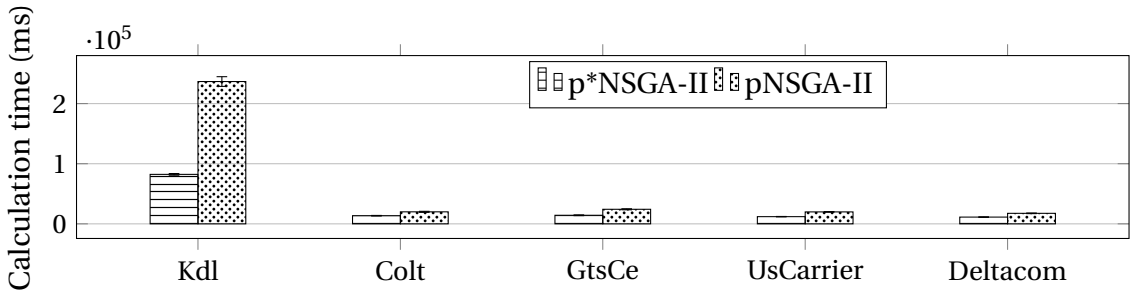


Figure 4.63 – Temps de calcul de p*NSGA-II and pNSGA-II

Le paramétrage des algorithmes et les réseaux testés sont pratiquement identiques à ceux utilisés pour le problème du routage. Ils sont présentés dans les tableaux 4.6 et 4.7

Nous considérons un VNF-FG avec 4 VNF connectées en série. Chaque VNF nécessite quatre types de ressources : CPU, mémoire, stockage et ressources radios. Nous supposons que la première VNF nécessite des ressources radios spécifiques alors que d'autres n'en ont pas besoin. De plus, le nombre de nœuds de l'infrastructure pouvant accueillir la première VNF est limité en raison de la situation géographique des utilisateurs. Pour simplifier, nous supposons que les hôtes de la première VNF et de la dernière VNF sont prédéfinis.

Pour le problème d'allocation VNF-FGs, la taille par défaut de la population (désignée par PS) est de 20 et le nombre de générations, désigné par (GS) est de 120. Les tailles de population pour le problème de routage (PR) sont 10 et 20 avec un nombre de générations respectif (GR) de 60 et 120. Comme nous l'avons déjà vu précédement, NDC-M coûte plus cher en temps de calcul tout en offrant des performances similaires mesurées avec l'hypervolume. Par conséquent, nous excluons NDC-M.

Les résultats de l'hypervolume sont normalisés par rapport à la borne supérieure présentée.

Comme nous l'avons vu aussi, l'efficacité de NSGA-II est inférieure d'environ 10% à NSGA-III et à SPEA2. Cependant, le temps de calcul très court de NSGA-II justifie sa mise en œuvre privilégiée. Les performances de p*NSGA-II dans le problème de placement des VNF-FGs sont illustrées dans la figure 4.63.

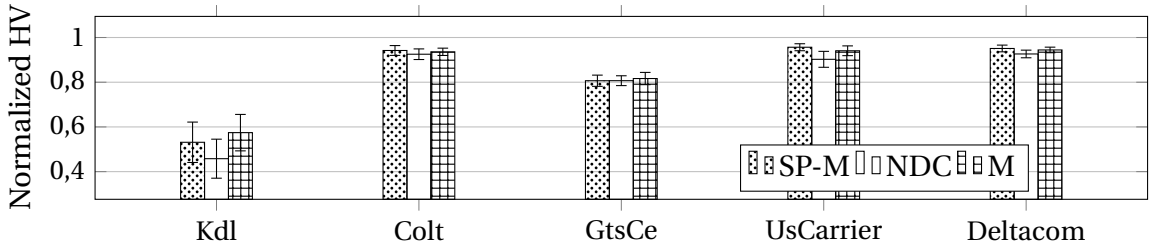
Dans la section sur le routage nous avons montré que la version parallèle p*NSGA-II surpasse pNSGA-II pour le problème de routage. L'écart de performance est même accru dans le problème de placement des VNF-FGs en raison de sa grande complexité de calcul.

Par exemple, le temps de calcul de pNSGA-II dans le scénario Kdl est deux fois plus long que celui de p*NSGA-II. Dans d'autres scénarios, les valeurs de temps de calcul de pNSGA-II sont supérieures de 50% à celles de p*NSGA-II.

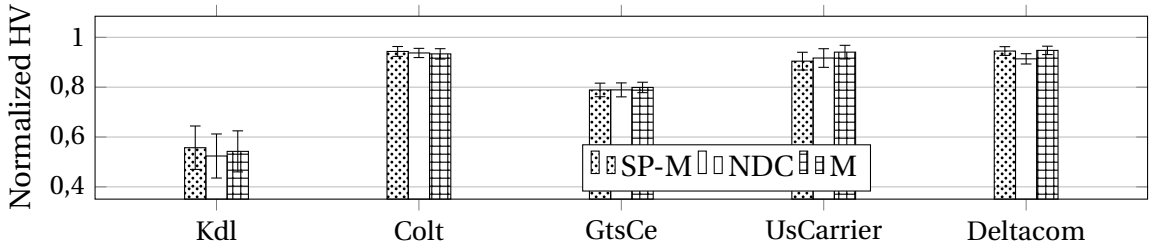
Comme nous l'avons déjà indiqué, l'algorithme GAR est utilisé pour déterminer l'ensemble des chemins non dominés entre deux hôtes d'une liaison virtuelle. Par conséquent, il est nécessaire d'examiner l'impact du problème de routage sur le problème d'allocation des VNF-FG.

L' hypervolume normalisé est calculé comme le rapport de l'hypervolume obtenu et de la borne supérieure introduite dans la section 15.

La Figure 4.64 présente l'HV normalisé pour différents types d'algorithmes de routage et les valeurs GR et PR. Pour l'HV normalisé, il existe des différences significatives entre NDC et les deux autres mécanismes lorsque GR et PR sont faibles. Cependant, cet écart est négligeable lorsque GR et PR sont réglés respectivement à 120 et 20. Les HV des trois scénarios : Colt, UsCarrier et Deltacom sont proches de la limite supérieure. Cependant,



(a) Normalized Hypervolume of VNF-FGs allocation problem with PR=10 and GR=60



(b) Normalized Hypervolume of VNF-FGs allocation problem with PR=20 and GR=120

Figure 4.64 – Hypervolume normalisé pour l'allocation de VNF-FGs

les hypervolumes de GtsCe et Kdl sont loin de la limite supérieure. C'est parce que la complexité de Kdl et GtsCe est plus élevée que les autres du fait du nombre de liens et de la longueur relativement courte des VNF-FGs. Les temps de calcul des scénarios sont présentés dans la Figure 4.65.

L'augmentation de la taille de la population et du nombre de générations augmente le temps de calcul. Même dans un scénario très complexe (par exemple Kdl), le temps de calcul est toujours inférieur à 100s dans les deux configurations.

Pour traiter les topologies très complexes, une approche évidente consiste à augmenter la taille de la population et le nombre de générations de l'AEMO, c'est-à-dire les paramètres PS et GS.

La Figure 4.66 montre les améliorations obtenues pour l'hypervolume pour des valeurs PS = 80 et GS = 480 dans les scénarios Kdl et GtsCe. Les hypervolumes des deux scénarios augmentent lorsque PS et GS augmentent. Bien qu'il y ait une légère augmentation dans le scénario GtsCe (autour de 10%), une augmentation remarquable est surtout observée dans le scénario Kdl.

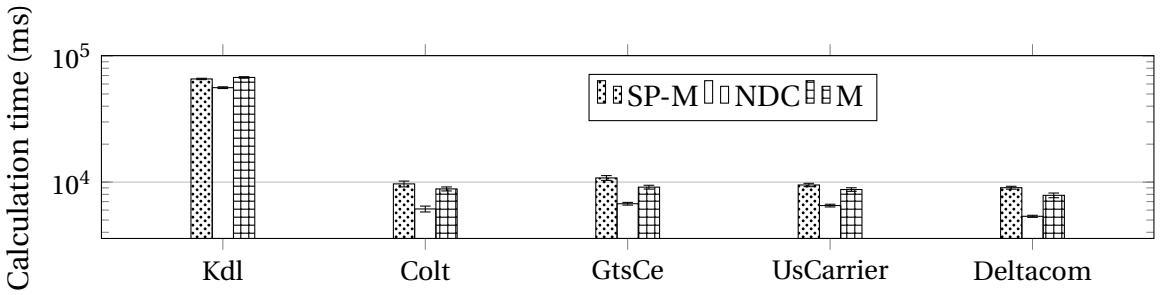
Le temps de calcul dans le scénario Kdl avec PS = 80 et GS = 480 est de l'ordre 200s à 300s.

Il faut remarquer que nous pouvons le réduire de manière significative en utilisant des plates-formes supportant le calcul parallèle qui sont disponibles dans les centres de données. Il est remarquable que l'approche proposée surpasse cependant les travaux existants en termes de temps d'exécution. Par exemple, il faut 40 secondes pour déterminer l'emplacement d'une chaîne 3-VNF avec une petite topologie d'infrastructure, comme celle de BT Europe (24 nœuds, 74 arcs), et 500 générations selon les travaux présentés par [KHEBBACHE et collab., 2018].

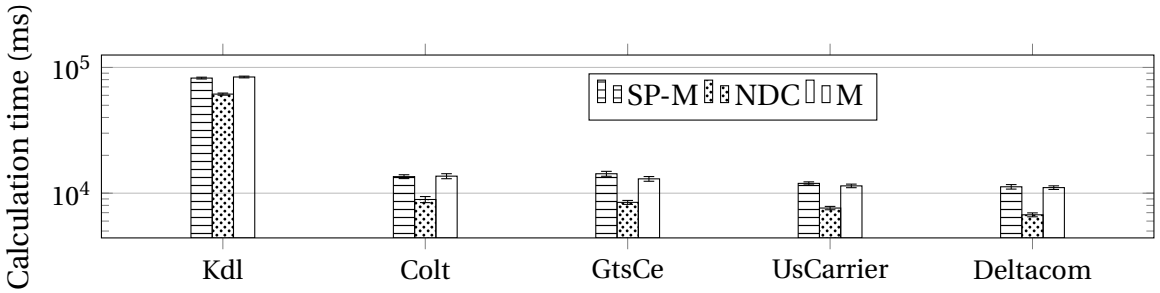
D'autre part, notre approche peut trouver un bon HV pour un réseau d'infrastructure beaucoup plus grand tel que GtsCe (149 nœuds, 386 liens), en moins de 20 secondes (avec 120 générations).

[LANGE et collab., 2017], ont étudié le placement d'une chaîne de 4 VNF sur un petit réseau de d'infrastructure, Internet2 (12 nœuds, 15 liens).

Il leur faut environ 2 secondes pour atteindre 90% de l'hypervolume. Cependant,



(a) Temps de calcul avec 10 individus et 60 générations



(b) Temps de calcul avec 20 individus et 120 generations

Figure 4.65 – Temps de calcul

nous obtenons de meilleures performances avec NHV = 90% dans un réseau beaucoup plus grand tel que Deltacom (113 nœuds, 322 liens) en 2 secondes.

Le taux de perte, le coût de déploiement et la latence obtenue pour (SP-M) sous différentes configurations de PS et GS dans le scénario Kdl sont illustrés dans la Fig. 4.67. En augmentant PS et GS, l'algorithme est capable d'améliorer à la fois la qualité et le nombre de solutions, et le NHV augmente comme indiqué sur la Fig. 4.66.

Différentes longueurs de VNF-FG Nous avons enfin effectué des simulations pour différentes longueurs de VNF-FG. Le but était de montrer comment les performances changent lorsque la complexité des VNF-FG augmente.

La figure 4.68 présente les performances obtenues dans le scénario Colt pour différents nombres de VNF (4, 6, et 8).

En ce qui concerne l' hypervolume normalisé, l'augmentation du nombre de VNFs entraîne sa dégradation. En raison de la contrainte de localisation des VNFs, la différence entre la solution trouvée et la solution optimale du problème de routage $\widehat{S}$ augmente.

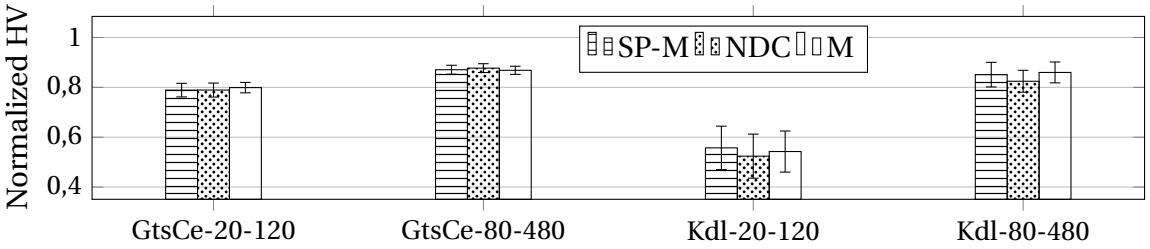
En outre, le nombre plus élevé de VNFs exige d'explorer un plus grand nombre de possibilités d'installation des VNFs.

En ce qui concerne le temps de calcul, la figure 4.68b montre que le temps de calcul est linéaire en fonction de la taille des VNFs. Elle confirme donc la scalabilité de l'approche proposée.

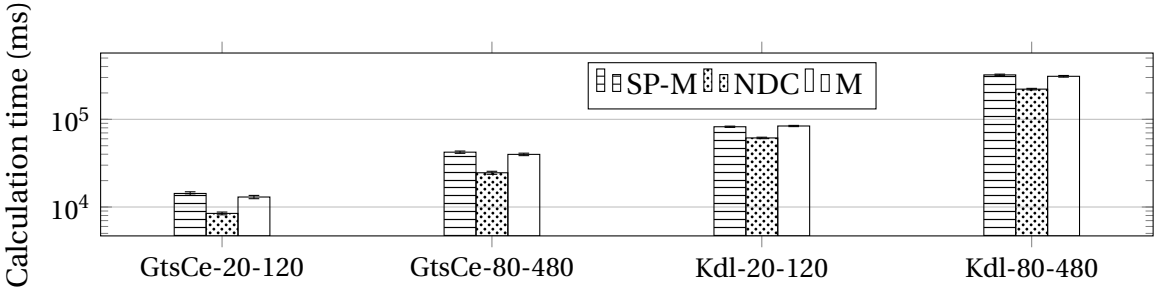
4.5.3 Conclusion sur le placement de VNF-FG

L'approche proposée peut être étendue au placement de plusieurs VNF-FG qui peut être vu comme le placement d'une seule VNF-FG en autorisant des discontinuités.

L'explosion combinatoire liée à l'augmentation du nombre de VNFs (la taille de l'espace du problème croît de manière exponentielle par rapport au nombre de VNFs) est atténuée par la mise en œuvre d'une méta-heuristique.



(a) Normalized HV 80-480



(b) Calculation time 20-120 vs. 80-480

Figure 4.66 – Performance 20-120 vs. 80-480

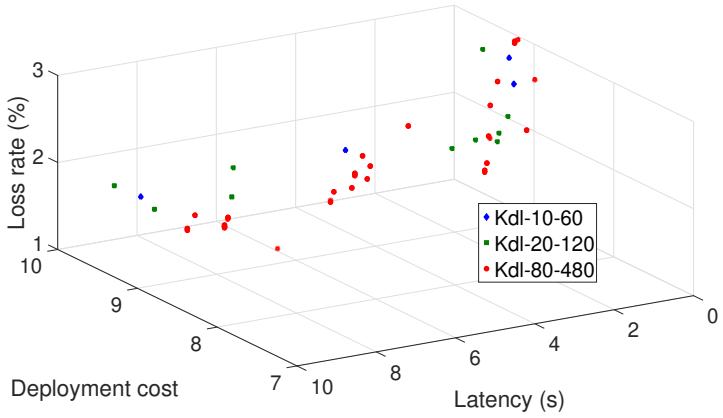
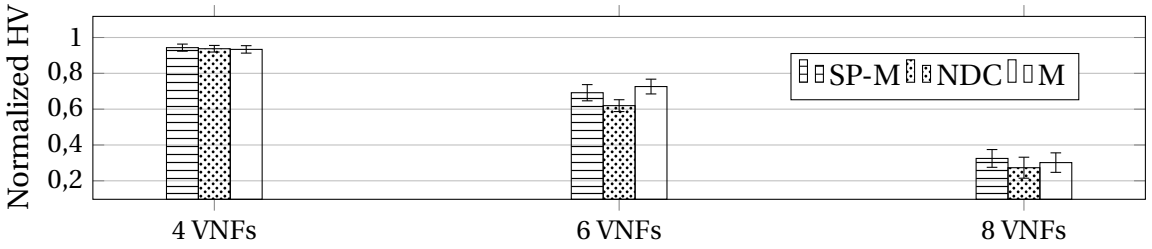


Figure 4.67 – Performances de GAVA dans le scénario Kdl

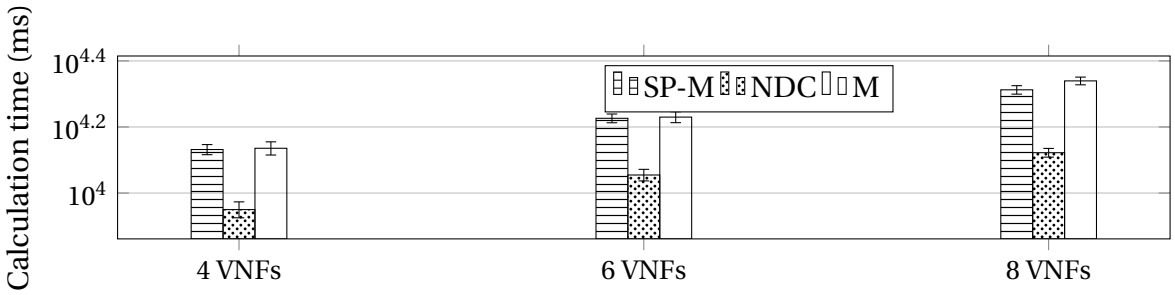
L'approche de placement dynamique que nous avons présentée pour le placement des contrôleurs peut être aussi étendue au placement de VNF-FGs de manière similaire pour traiter de scénario de placement dynamique comme dans l'approche de [OTOKURA et collab., 2016].

L'une des limitations de ce travail réside dans le fait que l'optimisation proposée est locale au service réseau et ne concerne que des métriques additives pour la partie réseau des VNFs. Cependant, les fonctions de coûts attachées aux liens et aux nœuds permettent d'obtenir des objectifs globaux.

La structure des AEMO permet aussi d'étendre à des objectifs supplémentaires globaux à l'infrastructure en s'appuyant sur les capacités multi-objectifs de NSGA-II et NSGA-III qui sont suffisamment flexibles pour permettre d'introduire de nouveaux objectifs.



(a) HV normalisé pour Colt 20-120 avec différents nombre de VNFs



(b) Calculation time Colt 20-120 under different numbers of VNFs

Figure 4.68 – Performance de Colt 20-120 avec différents nombres de VNFs

4.6 Conclusion sur le placement de services réseaux

Dans ce chapitre nous avons abordé différents problèmes de placement, le placement de contrôleurs, le placement de chemins et le placement de chaînes de VNFs. Pour le placement de contrôleurs nous avons proposé un algorithme très rapide basé sur une heuristique qui permet de réaliser un placement sur des instances de réseaux de taille relativement importante. Cette heuristique pourrait être améliorée et être étendue à d'autres métriques.

De manière à disposer d'un outil plus générique et multi-objectifs, nous avons utilisé les AEMO pour placer des contrôleurs de manière optimale en particulier par rapport à une métrique de connectivité.

Nous avons essayé d'étendre le concept afin de disposer d'un outil générique pour le placement de services réseaux, nous avons essayé d'utiliser les AEMO pour résoudre le problème du placement de VNF-FGs.

Les résultats obtenus montrent que les AEMO peuvent être très compétitifs en terme de temps de calculs contrairement à leur réputation. Il a fallu cependant pour cela introduire une certaine spécialisation en particulier avec l'introduction d'un opérateur de réparation, et de mécanismes d'optimisation locale ce qui nuit à la généricité. L'effet de cette spécialisation est aussi la réduction de la taille de la population.

On peut remarquer aussi que les opérateurs de recombinaison mis en œuvre n'apportent pas réellement d'améliorations aux performances des algorithmes.

Bibliography

AGRAMA, F. A. E.-M. 2011, «Linear projects scheduling using spreadsheets features», *Alexandria Engineering Journal*, vol. 50, nº 2, doi:https://doi.org/10.1016/j.aej.2011.01.018, p. 179 – 185, ISSN 1110-0168. URL `http://www.sciencedirect.com/science/article/pii/S1110016811000408`.

ANH QUANG, P. T., J. SANNER, C. MORIN et Y. HADJADJ-AOUL. 2018, «Multi-objective multi-constrained qos routing in large-scale networks: A genetic algorithm approach», dans *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, p. 55–60, doi:10.1109/SaCoNeT.2018.8585634.

ARTHUR, D. et S. VASSILVITSKII. 2007, «K-means++: The advantages of careful seeding», dans *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, ISBN 978-0-898716-24-5, p. 1027–1035.

AUGER, A., J. BADER, D. BROCKHOFF et E. ZITZLER. 2012, «Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications.», *Theor. Comput. Sci.*, vol. 425, p. 75–103.

BAR-ILAN, J. et D. PELEG. 1991a, *Approximation algorithms for selecting network centers*, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-540-47566-8, p. 343–354, doi:10.1007/BFb0028274. URL http://dx.doi.org/10.1007/BFb0028274.

BAR-ILAN, J. et D. PELEG. 1991b, «Approximation algorithms for selecting network centers (preliminary vesion).», dans *WADS*, *Lecture Notes in Computer Science*, vol. 519, édité par F. K. H. A. Dehne, J.-R. Sack et N. Santoro, Springer, ISBN 3-540-54343-0, p. 343–354.

BEINEKE, L., O. OELLERMANN et R. E. PIPPERT. 2002a, «The average connectivity of a graph», vol. 252, p. 31–45.

BEINEKE, L. W., O. R. OELLERMANN et R. E. PIPPERT. 2002b, «The average connectivity of a graph», *Discrete Mathematics*, vol. 252, nº 1, doi:https://doi.org/10.1016/S0012-365X(01)00180-7, p. 31 – 45, ISSN 0012-365X. URL http://www.sciencedirect.com/science/article/pii/S0012365X01001807.

BHESDADIYA, R. H., I. N. TRIVEDI, P. JANGIR, N. JANGIR et A. KUMAR. 2016, «An nsga-iii algorithm for solving multi-objective economic/environmental dispatch problem», *Cogent Engineering*, vol. 3, nº 1, doi:10.1080/23311916.2016.1269383. URL http://doi.org/10.1080/23311916.2016.1269383.

BRANDES, U. 2001, «A faster algorithm for betweenness centrality», *Journal of Mathematical Sociology*, vol. 25, p. 163–177.

BRON, C. et J. KERBOSCH. 1973, «Algorithm 457: Finding all cliques of an undirected graph», *Commun. ACM*, vol. 16, nº 9, doi:10.1145/362342.362367, p. 575–577, ISSN 0001-0782. URL http://doi.acm.org/10.1145/362342.362367.

CIRO, G. C., F. DUGARDIN, F. YALAOUI et R. KELLY. 2016, «A nsga-ii and nsga-iii comparison for solving an open shop scheduling problem with resource constraints», *IFAC-PapersOnLine*, vol. 49, nº 12, doi:https://doi.org/10.1016/j.ifacol.2016.07.690, p. 1272 – 1277, ISSN 2405-8963. URL http://www.sciencedirect.com/science/article/pii/S2405896316309661, 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.

CROWCROFT, J. 1996, «Quality-of-service routing for supporting multimedia applications», *IEEE Journal on Selected Areas in Communications*, vol. 14, nº 7, doi:10.1109/49.536364, p. 1228–1234, ISSN 0733-8716.

DEB, K., A. PRATAP, S. AGARWAL et T. MEYARIVAN. 2002, «A fast and elitist multiobjective genetic algorithm: Nsga-ii», *IEEE Transactions on Evolutionary Computation*, vol. 6, nº 2, doi:10.1109/4235.996017, p. 182–197, ISSN 1089-778X.

EDMONDS, J. et R. M. KARP. 1972, «Theoretical improvements in algorithmic efficiency for network flow problems», *J. ACM*, vol. 19, nº 2, doi:10.1145/321694.321699, p. 248–264, ISSN 0004-5411. URL http://doi.acm.org/10.1145/321694.321699.

ESTER, M., H.-P. KRIEGEL, J. SANDER et X. XU. 1996, «A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise», dans *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, AAAI Press, p. 226–231. URL http://dl.acm.org/citation.cfm?id=3001460.3001507.

GAREY, M. R. et D. S. JOHNSON. 1990, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, ISBN 0716710455.

HOLZINGER, A., D. BLANCHARD, M. BLOICE, K. HOLZINGER, V. PALADE et R. RABADAN. 2014, «Darwin, lamarck, or baldwin: Applying evolutionary algorithms to machine learning techniques», dans *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 02*, WI-IAT '14, IEEE Computer Society, Washington, DC, USA, ISBN 978-1-4799-4143-8, p. 449–453, doi:10.1109/WI-IAT.2014.132. URL http://dx.doi.org/10.1109/WI-IAT.2014.132.

JAIN, S., A. KUMAR, S. MANDAL, J. ONG, L. POUTIEVSKI, A. SINGH, S. VENKATA, J. WANDERER, J. ZHOU, M. ZHU, J. ZOLLA, U. HÖLZLE, S. STUART et A. VAHDAT. 2013, «B4: Experience with a globally-deployed software defined wan», *SIGCOMM Comput. Commun. Rev.*, vol. 43, nº 4.

KARP, R. 1972, «Reducibility among combinatorial problems», dans *Complexity of Computer Computations*, édité par R. Miller et J. Thatcher, Plenum Press, p. 85–103.

KARP, R. M. dans *50 Years of Integer Programming*, édité par M. Junger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi et L. A. Wolsey.

KARP, R. M. 2010, «Reducibility among combinatorial problems», dans *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, p. 219–241.

KASHTAN, N., E. NOOR et U. ALON. 2007, «Varying environments can speed up evolution», *Proceedings of the National Academy of Sciences*, vol. 104, nº 34, doi:10.1073/pnas.0611630104, p. 13 711–13 716, ISSN 0027-8424. URL http://www.pnas.org/content/104/34/13711.

KHEBBACHE, S., M. HADJI et D. ZEGHLACHE. 2018, «A multi-objective non-dominated sorting genetic algorithm for vnf chains placement», dans *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, ISSN 2331-9860, p. 1–4, doi:10.1109/CCNC.2018.8319250.

LANGE, S., A. GRIGORJEW, T. ZINNER, P. TRAN-GIA et M. JARSCHEL. 2017, «A multi-objective heuristic for the optimization of virtual network function chain placement», dans *2017 29th International Teletraffic Congress (ITC 29)*, vol. 1, p. 152–160, doi:10.23919/ITC.2017.8064351.

NEBRO, A. J., J. J. DURILLO et M. VERGNE. 2015, «Redesigning the jmetal multi-objective optimization framework», dans *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, ACM, New York, NY, USA, ISBN 978-1-4503-3488-4, p. 1093–1100, doi:10.1145/2739482.2768462. URL http://doi.acm.org/10.1145/2739482.2768462.

NETHERCOTE, N., P. J. STUCKEY, R. BECKET, S. BRAND, G. J. DUCK et G. TACK. 2007a, «Minizinc: Towards a standard cp modelling language.», dans *CP*, *Lecture Notes in Computer Science*, vol. 4741, édité par C. Bessiere, Springer, ISBN 978-3-540-74969-1, p. 529–543. URL http://dblp.uni-trier.de/db/conf/cp/cp2007.html#NethercoteSBBDT07.

NETHERCOTE, N., P. J. STUCKEY, R. BECKET, S. BRAND, G. J. DUCK et G. TACK. 2007b, «Minizinc: Towards a standard cp modelling language», dans *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, CP'07, ISBN 978-3-540-74969-1, p. 529–543.

OTOKURA, M., K. LEIBNITZ, Y. KOIZUMI, D. KOMINAMI, T. SHIMOKAWA et M. MURATA. 2016, «Application of evolutionary mechanism to dynamic virtual network function placement», dans *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, p. 1–6, doi:10.1109/ICNP.2016.7784475.

PHAM TRAN ANH QUANG JEAN-MICHEL SANNER, C. M. Y. H.-A. 2018, «Virtual network function forwarding graph embedding: A genetic algorithm approach», *Saconet speical issues*.

QUANG, P. T. A., K. PIAMRAT, K. D. SINGH et C. VIHO. 2017, «Video streaming over ad hoc networks: A qoe-based optimal routing solution», *IEEE Transactions on Vehicular Technology*, vol. 66, nº 2, doi:10.1109/TVT.2016.2552041, p. 1533–1546, ISSN 0018-9545.

RODRIGUEZ-COLINA, E., D. GIL-LEYVA, J. L. MARZO et V. M. RAMOSR. 2014, «A bit error rate analysis for tcp traffic over parallel free space photonics», *Telecommunication Systems*, vol. 56, nº 4, p. 455–466.

SAKAROVITCH, M. 1984, *Optimisation combinatoire: Programmation discrète*, Collection Enseignement des sciences, Hermann, ISBN 9782705659769. URL https://books.google.fr/books?id=uUrvAAAAMAAJ.

SHARMA, D. et P. COLLET. 2010, «Gpgpu-compatible archive based stochastic ranking evolutionary algorithm (g-asrea) for multi-objective optimization», dans *Parallel Problem Solving from Nature, PPSN XI*, édité par R. Schaefer, C. Cotta, J. Kołodziej et G. Rudolph, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-642-15871-1, p. 111–120.

STUCKEY, P. J., T. FEYDY, A. SCHUTT, G. TACK et J. FISCHER. 2014, «The minizinc challenge 2008-2013», *AI Magazine*, vol. 35, nº 2, p. 55–60.

TACK, G. 2009, *Constraint Propagation - Models, Techniques, Implementation*, phdthesis, Saarland University, Germany.

VAN MIEGHEM, P. et F. A. KUIPERS. 2004, «Concepts of exact qos routing algorithms», *IEEE/ACM Transactions on Networking*, vol. 12, nº 5, doi:10.1109/TNET.2004.836112, p. 851–864, ISSN 1063-6692.

ZITZLER, E., M. LAUMANNS et L. THIELE. 2001, «SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization», dans *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, édité par K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou et T. Fogarty, International Center for Numerical Methods in Engineering, Athens, Greece, p. 95–100.

Chapitre 5

Conclusion et perspectives

“ 'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves, And
the mome raths outgrabe. ”

Lewis Caroll

Sommaire

5.1 Conclusion

5.1.1 Architecture SDN

Le modèle “DICES” qui a fait l’objet de notre première contribution dans le volet architecture de cette thèse est assez éloigné des orientations prises depuis quelques années pour les architectures des réseaux de télécommunication dans le cadre des paradigmes SDN et ETSI-NFV.

Pour le définir nous nous sommes appuyés sur les discussions des chercheurs du domaine autour de la pertinence des architectures SDN centralisées.

L’orientation prise par les opérateurs Telco pour faire évoluer leur réseau consiste depuis 3 à 4 ans à virtualiser les fonctions réseaux sur des serveurs COTS et si possible dans des datacenters.

C’est la dynamique apportée par la virtualisation, et elle est fortement calquée sur le développement des technologies de déport de services informatiques dans les datacenters. Elle reprend les mêmes modèles en faisant l’hypothèse que les services réseaux pourront être portés sur ces serveurs exactement comme n’importe quelle applications informatiques sous formes de fonctions réseaux virtualisées.

Si cette tendance semble bien engagée elle n’est pas encore réellement déployée aujourd’hui car elle implique des changements très importants d’un point de vue technologique et humain, et en terme d’organisation. Elle rencontre aussi pour l’instant des difficultés techniques concrètes et des verrous qui tournent autour de la fiabilité, de la consommation énergétique et des performances qui sont des points clés pour des services réseaux destinés à répondre aux besoins des applications.

La compatibilité entre l’approche purement logicielle et les contraintes des services réseaux ne va pas de soi. Nous ne doutons cependant pas qu’elle aboutira compte tenu des moyens mis en œuvre. Il est possible aussi, cependant, que cette mise en œuvre reste longtemps restreinte à quelques cas d’usages privilégiés.

Un peu à contre courant, nous avons proposé dans notre travail de définir des nœuds intelligents (fortement programmables) dont les performances pourraient dépendre du choix des architectures sous-jacentes (hardware programmable, nœuds virtualisés sur des machines virtuelles..).

La programmation serait réalisée via un protocole plus flexible et plus riche que le protocole OF. Ce protocole intégrerait des fonctions de contrôle et de gestion et aussi de programmation du plan de données.

Les nœuds réseaux disposeraient d’une architecture générique incluant une partie contrôle et une partie plan de données programmables.

Cette approche prend le parti que les performances demandées aux services réseaux auront toujours un temps d’avance par rapport aux performances obtenues pour la virtualisation.

Cette approche a aussi bien sûr été identifiée depuis longtemps par des industriels des télécommunications comme Cisco qui ont enrichi leurs équipements pour permettre une certaine programmabilité.

Elle existe aussi en normalisation par exemple avec le standard I2RS définit à l’IETF qui a pour objectif de standardiser une interface de programmation vers les équipements routeurs. Ce standard a d’ailleurs été imaginé pour proposer une alternative à OF.

Nous avons privilégié l’idée de nœuds réseaux intelligents et programmables qu’ils soient physiques ou virtuels, plutôt que la manipulation de machines virtuelles qui supportent de très volumineux fichiers binaires embarquant les services et les fonctions du

réseau.

La preuve de concept que nous avons implémenté illustre la faisabilité d'un plan de contrôle SDN aux fonctionnalités étendues par rapport à OF.

En nous appuyant sur l'architecture Click nous avons montré qu'on pouvait obtenir des fonctions réseaux facilement évolutives, adaptatives et reconfigurables dynamiquement avec une approche beaucoup plus flexible et riche que celle d'OF.

D'autres travaux s'appuyant sur l'architecture Click ont montré l'intérêt de cette approche avec la modélisation de fonctions réseaux plus complexes comme les Broadband Network Gateway (BNG).

5.1.2 Placement des services réseaux

Dans cet autre volet de ce travail de thèse, notre démarche a été de partir de la problématique du placement de contrôleurs dans une infrastructure physique ou virtuelle puis d'étendre la problématique au placement de services réseaux dans un contexte de virtualisation. Nous avons ainsi abordé le placement de chemins et le placement de graphe d'acheminement des fonctions réseaux virtualisées (VNFFG – VNF Forwarding Graph).

Nous sommes cependant restés dans une approche assez académique dans la mesure où nous n'avons pas réalisé d'implémentation effective dans des outils adaptés comme PCE dans les réseaux MPLS pour l'optimisation du routage, ou Watcher pour le placement de fonctions réseaux hébergées dans des machines virtuelles ou encore Openstack et ONAP-OF pour des infrastructures de déploiement de services réseaux virtualisés.

Les travaux réalisés autour du placement de chemins en collaboration avec M. Quang Phan tran han sont cependant portables dans le contexte d'un PCE développé à BCOM par notre collègue M. Cédric Morin.

Nous avons utilisé pour nos évaluations des graphes de réseaux WAN tirés de la base de données zoo-topology. Ces topologies présentent l'intérêt d'être très diverses mais elles ne correspondent pas nécessairement aux topologies des infrastructures virtualisées du paradigme NFV. Il aurait été intéressant de travailler à partir d'une topologie d'une infrastructure d'opérateur comme celle d'Orange.

Nous avons essayé d'initier ce travail sur la topologie du réseau Orange national, le Réseau Backbone de Collecte Internet (RBCI). Cette topologie est composée de plusieurs milliers de routeurs, organisés en trois sous-structures: le réseau d'accès, le réseau de collecte et le réseau cœur.

Cette infrastructure réseau est encore cependant très loin d'être SDNisée et encore moins virtualisée. En effet il s'agit avant tout d'une infrastructure de routage. Définir des règles d'ingénierie permettant de déployer des contrôleurs dans un tel contexte aurait pu être le sujet d'un travail d'une thèse.

Mais, d'un autre coté, il s'avère que la recherche sur le déploiement de contrôleurs SDNs dans les infrastructures ne s'est pas orienté dans le sens de la prise en charge de fonctions de base comme le routage mais plutôt comme un moyen d'établir des chemins afin de déployer des services réseaux organisés sous forme de VNF-FGs dans le contexte de la virtualisation.

Dans le contexte NFV, les topologies d'infrastructure de virtualisation visées par les opérateurs ne sont pas encore réellement déployées sinon à titre expérimental. Elles seront à priori structurées avec quelques centaines de nœuds de petites capacités de stockage et de calcul en périphérie, puis de quelques dizaines de noeuds régionaux de capacité moyenne pour le plan de données avec une couverture d'environ 200 km de rayon, et enfin quelques nœuds de capacités plus importantes sous la forme de datacenters na-

tionaux avec des fonctions de contrôle et de gestion. Tout cela est complété par quelques datacenters internationaux.

Ces topologies cibles ne sont pas finalement si éloignées de certaines topologies WAN tirées de la base de données zoo-topologie. Nous avons vérifié que les algorithmes que nous avons développés sont compatibles avec le dimensionnement de telles infrastructures. L'heuristique hiérarchique est rapide même avec plusieurs milliers de nœuds et les algorithmes évolutionnaires peuvent être optimisés et accélérés en introduisant du parallélisme et du code optimisé.

Nous avons aussi vérifié le comportement de nos algorithmes sur des topologies analogues de petites tailles et la cohérence des placements obtenus.

Par ailleurs nos expérimentations et travaux autour des AEs utilisés pour résoudre des problèmes de placement et d'allocation de ressources, ont permis de dégager un certain nombre de constats et de principes pour déterminer les principes à prendre en compte dans la définition d'un solveur générique multi-objectifs pour les problèmes de placement de ressources. Nous les énumérons ci-dessous.

- Un AE avec un simple opérateur de mutation est parfaitement viable,
- La définition d'un opérateur de recombinaison est dépendante du problème et peut être délicate. Si il est efficace il peut cependant accélérer fortement la convergence de l'algorithme,
- La définition d'un opérateur d'optimisation local est dépendante du problème et peut induire des temps de calcul supplémentaires mais accélérer fortement la convergence,
- Limiter la population à des individus viables est plus efficace mais moins générique car pour obtenir des individus viables à chaque étape il peut être nécessaire de spécialiser l'algorithme en fonction du problème posé et de définir par exemple des opérateurs de variation adaptés,
- La notion d'opérateur de réparation permet d'utiliser des modèles très génériques sous forme de mots binaires mais le réparateur doit être spécialisé en fonction du problème à résoudre. Il est en fait dépendant du problème. De plus, le réparateur doit être activé à chaque itération ce qui peut être coûteux en temps de calcul. Cependant le réparateur peut-être aussi parallélisé,
- Une structure de données qui garantirait des individus viables à chaque étape est idéale, l'utilisation de mots binaires et d'algorithmes génétiques standards n'est donc pas adaptée,
- La structure de données la plus adaptée pour le problème du placement des contrôleurs est une liste de groupes,
- La structure de données la plus adaptée pour le problème du routage est une liste d'arcs du graphe ou de paires de noeuds sources-destinations,
- La structure de données pour le problème du placement de VNF-FGs est le chemin adopté pour chaque VNF-FG (voir item précédent) et la liste des nœuds hébergeurs pour chaque VNF,
- Le problème du routage nécessite des opérateurs de variations adaptés pour maintenir la viabilité des individus,

- Les algorithmes NSGA II et III sont performants mais ne peuvent pas être complètement parallélisés,
- La possibilité de paralléliser les algorithmes évolutionnaires doit être exploitée au maximum. L'implémentation sur des cartes GPGPU par exemple aurait un potentiel très important,
- Le parallélisme peut être aussi une manière de se passer d'opérateur de recombinaison en manipulant des populations de taille importante et en échantillonnant ainsi mieux l'espace de recherche,
- Les algorithmes évolutionnaires peuvent être adaptés pour gérer des scénarios de placement en ligne dynamiques.

5.2 Perspectives

5.2.1 Architecture SDN

La preuve de concepts que nous avons implémentée ne traite pas cependant d'autres aspects de notre proposition DICES qu'il faudrait étudier. En particulier, on peut citer:

1. La spécification détaillée du protocole qui se substituerait à OF,
2. La spécification détaillée d'un composant Click générique intégrant une fonction de contrôle,
3. La spécification de l'approche de modélisation des applications réseaux sous forme de réseaux de Petri,
4. La problématique de la complétude du modèle de décomposition d'un service réseau (intégrant le plan de contrôle et le plan de données),
5. La spécification d'une couche d'abstraction permettant de composer des services réseaux de manière semi-automatisée à partir de composants réseaux élémentaires, en utilisant un langage de plus haut niveau,
6. L'architecture détaillée d'un élément réseau type adapté à ce type d'approches qui met à disposition des fonctions du plan de contrôle et du plan de données sous forme d'API.

Pour l'item 2, nous avons introduit dans le cadre de la preuve de concepts des éléments Click "contrôleurs" qui permettent d'introduire une notion de contrôle dans la logique Click. Ces éléments permettent de réagir à des évènements, d'apporter une intelligence dans le traitement et de faire évoluer les configurations via une hiérarchie de contrôleurs. Mais il faudrait spécifier un composant type intégrant cet élément de contrôle.

Pour l'item 3, les principes de la modélisation de tous services réseaux avec des réseaux de Petri restent à préciser. Une approche évidente consisterait à mapper simplement les jetons qui représentent de la ressource avec les paquets IP, cependant, d'autres approches sont possibles. Les jetons peuvent représenter d'autres types de ressources, des flots par exemple, qui sont plus généraux que de simples paquets IP, ou des chemins

dans le graphe du réseau pour traiter des services réseaux où la fonction de base est l'attribution d'un chemin.

Un aspect très intéressant des réseaux de Petri est la possibilité de leur enrichissement. Les réseaux de Petri colorés pourraient permettre par exemple de modéliser des VNF-FGs qui intégreraient à la fois le plan de données et le plan de contrôle comme par exemple un modèle de réseau mobile virtualisé, en distinguant deux types de ressources IP. L'introduction des réseaux à file d'attente permettrait de modéliser les aspects performances attendues de ces modèles.

Pour l'item 4, l'idée est que l'on doit pouvoir décrire n'importe quel service réseau à partir de l'outil de modélisation. Cela suppose que la librairie de composants élémentaires et la syntaxe d'assemblage (modèle à base de graphe standard de Click ou réseau de Petri) permettent d'assurer cette propriété.

Enfin le choix de Click comme moyen d'implémentation peut être discuté. D'autres alternatives sont en effet possibles comme eBPF ou VPP et une analyse comparative fine est nécessaire mais qui n'est pas forcément simple.

5.2.2 Placement de services réseaux

Des constats que nous avons effectué, nous retenons les principes suivants pour un solveur générique utilisant les AEMOs. L'intérêt des AEMOs réside dans leur flexibilité et leur capacité à résoudre une grande variété de problèmes.

Structure de données Le tableau 5.2.2, résume les structures de données qui seraient mise en oeuvre pour un tel outil.

Problème	Modèle
Placement de contrôleurs:	liste de groupes (ensembles de nœuds du réseau)
Placement de chemins:	liste des arcs (paires de sources et destinations)
Placement de VNFs:	liste des noeuds supportant les VNFs
Placement de VNF-FGs:	les deux modèles précédents

Framework d'AE Pour être aussi générique que possible l'algorithme ne doit pas inclure d'opérateurs de recombinaison, ni d'optimisation locale ou tout au moins ceux-ci doivent être optionnels et activables ou désactivables en fonction des problèmes traités car si ils existent, ils sont dépendants du problème.

L'opérateur de mutation ne doit pas créer des individus non viables pour éviter d'introduire un opérateur de réparation. Il y a une difficulté dans le cas du calcul de chemins.

Dans le cas du calcul de chemins qui joue un rôle aussi dans le placement de VNF-FG, une difficulté réside dans la définition d'opérateurs de mutation et surtout de recombinaison efficaces. Une manière de faire pourrait être d'utiliser des techniques mises en œuvre pour le problème du voyageur de commerce (2.4.3). Même si le problème n'est pas exactement identique, ces méthodes doivent pouvoir être transposées et elles sont très efficaces.

Nous proposons enfin de mettre en œuvre l'algorithme MOEA parallèle proposé par [SHARMA et COLLET, 2010], il promet des gains de temps de calcul très élevés ce qui permet des populations de grande taille avec des performances identiques à NSGA-II. Il peut être implémenté sur des cartes GPGPU.

En prenant davantage de hauteur, les algorithmes évolutionnaires sont un des multiples résultats de l'approche bio-mimétique qui consiste à s'inspirer du vivant pour développer des techniques de résolution de problèmes. Nous avons pu constater dans nos travaux à quel point il peut être nécessaire de revenir vers les approches de référence de la biologie pour comprendre les mécanismes qui s'en inspirent. Nous pensons que ce genre d'approche peut être encore et toujours très fructueuse dans le domaine des réseaux de télécommunications et aussi plus largement, en explorant différentes idées ou simplement pour améliorer des techniques déjà existantes comme les AE.

En effet il reste des pistes à explorer sur le paramétrage, ou pour mettre en œuvre des combinaisons avec les techniques d'apprentissage, ou bien même pour en imaginer de nouvelles, en partant de modèles récents de l'évolution expérimentés par les biologistes.

À l'heure des résultats phénoménaux obtenus à partir des techniques d'apprentissages profonds, une approche basée sur les algorithmes évolutionnaires peut paraître quelque peu périmée. Il nous semble intéressant de souligner au contraire les liens qui peuvent exister entre évolution et apprentissage. Ces idées sont mises en valeur dans l'ouvrage de [VALIANT, 2013] intitulé: "Probably Approximately Correct". Dans cet ouvrage l'auteur montre que l'évolution peut être vue comme une technique d'apprentissage PAC de type requête statistique (*statistical query*) et s'appuie en particulier sur les travaux de [FELDMAN, 2008].

Dans ce sens, dans la continuité de la thèse, nous avons initié dans le cadre de B-COM une collaboration avec le laboratoire ECOBIO de Rennes I avec l'idée d'explorer différentes pistes pour le problème du placement de ressources, et en particulier les trois suivantes:

- les modèles de l'évolution du vivant pour les algorithmes évolutionnaires et l'apprentissage appliqués à l'ingéniérie des réseaux,
- les modèles de fonctionnement des écosystèmes pour les mécanismes d'allocation de ressources avec la notion de capacité de charge (*Carrying capacity*),
- l'inférence de structures ADN ou d'arbres phylogénétiques à partir de données parcellaires ou incomplète pour le monitoring des infrastructures virtualisées.

Bibliography

FELDMAN, V. 2008, «Evolvability from learning algorithms», dans *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, ACM, New York, NY, USA, ISBN 978-1-60558-047-0, p. 619–628, doi:10.1145/1374376.1374465. URL http://doi.acm.org/10.1145/1374376.1374465.

SHARMA, D. et P. COLLET. 2010, «Gpgpu-compatible archive based stochastic ranking evolutionary algorithm (g-asrea) for multi-objective optimization», dans *Parallel Problem Solving from Nature, PPSN XI*, édité par R. Schaefer, C. Cotta, J. Kołodziej et G. Rudolph, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-642-15871-1, p. 111–120.

VALIANT, L. 2013, *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*, Basic Books, Inc., New York, NY, USA, ISBN 0465032710, 9780465032716.

Abréviations

AE	Algorithme évolutionnaire
AEMO	Algorithme évolutionnaire multi-objectifs
AE	Algorithme évolutionnaire
AEMO	Algorithme évolutionnaire Multi-Objectifs
AG	Algorithme Génétique
BNG	Broadband Network Gateway
CE	Control Element
CAP	Consistency Availability, Partition tolerance
CMA-ES	Covarariance Matrix Adaptation Evolution Strategy
COTS	Commercial Of The Shelf
DPI	Deep Packet Inspection
FE	Forwarding Element
FG	Forwarding Graph
GPU	Graphic Processor Unit
GPGPU	General Purpose Graphic Processing Unit
IP	Internet Protocol
MANO	Management And Network Orchestration
MPLS	Multiprotocol Label Switching
NFV	Network Function Virtualisation
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
OF	OpenFlow
ONAP	Open Network Automation Platform
ONAP-OF	Open Network Automation Platform Optimisation Framework
PAC	Probability Approximately Correct
PBN	Policy Based Networking
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PoC	Proof of Concept
RBCI	Réseau Backbone de Collecte Internet
SAMCRA	Self Adaptative MultiOjective Constraint Routing Algorithm
SDN	Software Defined Network
SD-WAN	Software Defined WAN
SLA	Service Level Agrement
SOF	OpenFlow Switch
SPOF	Single Point Of Failure
TSP	Traveller Salesman Problem
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNE	Virtual Network Embedding
VNF-C	Virtual Network Functions Components
VNF	Virtual Network Function
VNF-FG	VNF Forwarding Graph
VNFM	VNF Manager
WIM	WLAN Infrastructure Manager

Appendix A

Annexes

A.1 Notions théoriques sur la connectivité

Pré-requis sur la arc-connectivité et la nœud-connectivité dans un graphe Soit un graphe G(V, E) ou V est l'ensemble des nœuds du graphe et E l'ensemble des arcs du graphe

On définit la arc-connectivité $\lambda(u, v)$ entre deux nœuds u, v d'un graphe G comme le nombre minimum d'arcs qu'il faut enlever pour déconnecter u de v, c'est à dire de faire en sorte qu'il n'existe plus de chemin pour relier u et v.

Dans le cas d'un graphe orienté on peut avoir $\lambda(u, v) \neq \lambda(v, y)$, dans le cas d'un graphe non orienté le sens ne compte pas.

La définition de la nœud-connectivité est presque identique, on définit $\lambda'(u, v)$ entre deux nœuds u, v d'un graphe G comme le nombre minimum de nœuds qu'il faut enlever pour déconnecter u de v.

Ces deux définitions peuvent être généralisées aux graphes et à leur déconnexion en plusieurs composantes connexes. En effet, déconnecter deux nœuds quelconque d'un graphe, revient nécessairement à déconnecter le graphe. Et si deux nœuds sont déconnectés, il existe nécessairement au moins deux composantes connexes.

On définit par conséquent la k-connectivité $\lambda(G)$ d'un graphe comme le minimum de nœuds ou d'arcs qu'il faut enlever pour déconnecter le graphe en plusieurs composantes connexes. compte tenu des remarques précédentes, on a la relation A.1:

$$\lambda(G) = \min(\lambda(v, w)),\ \forall (v, w) \in G \text{ avec } (v, w) \neq (w, v) \tag{A.1}$$

Si G est un graphe orienté on considère les deux sens, c'est à dire que l'on distingue pour une paire de nœuds (v, w) la paire (v, w) de la paire (w, v), tandis que si G est non orienté on ne les distingue pas.

Enfin, de la même manière que pour les paires de nœuds d'un graphe, on distingue pour les graphes la k-arc-connectivité et la k-nœud-connectivité.

Ensemble de coupe On appelle ensemble de coupe l'ensemble des arcs qu'il faut retirer pour déconnecter deux nœuds.

On appelle ensemble de coupe minimum, l'ensemble d'arcs de cardinal minimum qu'il faut enlever pour déconnecter les deux nœuds.

Cette définition peut aussi être transposée à la nœud-connectivité en considérant l'ensemble des nœuds de cardinal minimum qu'il faut enlever pour déconnecter les deux nœuds.

Le théorème de Menger peut être formulé de manière presque identique pour la arc-connectivité et pour la nœud-connectivité. Il permet de relier la notion d'ensemble de coupe minimum à la notion de connectivité. Nous rappelons simplement un corollaire de ce théorème:

Corollaire 1 *Soit* G *un graphe fini orienté ou non orienté et deux sommets distincts x et y. soit c_k le cardinal de l'ensemble de coupe minimum de x et y (c'est à dire le nombre minimal d'arcs qui enlevés déconnectent x et y). Alors c_k est égal au nombre maximal de chemins indépendants (sans arcs communs aux deux chemins) de x à y.*

c'est à dire qu'il existe k chemins distincts si et seulement si il existe un ensemble de cardinal k.

Pour la nœud-connectivité il faut simplement exclure les cas ou s et t sont adjacents (dans ce cas la nœud-connectivité vaut 0), et la formulation est la même en remplaçant le mot “arc” par le mot "nœud".

On a aussi le corollaire suivant qui se déduit directement et qui s'applique à la totalité du graphe G. Il permet de définir précisément la k-arc-connectivité d'un graphe. la k-nœud-connectivité se déduit directement en remplacant arcs par nœuds et en excluant les nœuds extrémités.

Corollaire 2 *Soit un graphe non orienté* G*, avec au moins deux nœuds, le graphe est dit k-arc-connecté si et seulement si pour chaque paire s et t de* V(G) *avec $s \neq t$, il existe k chemins indépendants de s à t.*

Différence entre k-arc-connectivité et k-nœud-connectivité Nous avons eu dans notre travail quelques difficultés à faire la distinction entre arc-connectivité et nœud-connectivité dans un graphe.

En pratique, on peut simplement remarquer que si deux nœuds sont adjacents, la arc-connectivité doit prendre en compte l'arc reliant les deux nœuds tandis que la nœuds ne peut pas être calculée, puisqu'on ne peut pas enlever de nœuds pour les déconnecter.

Le schéma A.1 permet de voir sur un exemple simple qu'il n'est pas si évident de faire un lien entre arc-connectivité et nœuds-connectivité entre deux nœuds u et v.

On remarque dans ce schéma qu'il faut enlever deux arcs pour déconnecter les nœuds u et v et que cela correspond à deux chemins indépendants qui ne partagent pas d'arcs communs) entre u et v. Tandis qu'en enlevant le nœud 3 on déconnecte u et v, mais il n'y a qu'un seul chemin indépendant entre u et v (qui ne partagent pas de nœuds communs).

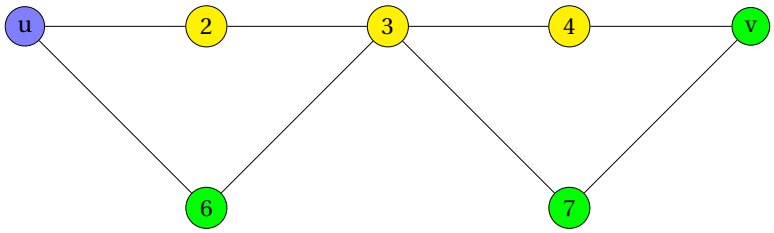


Figure A.1 – Exemple simple

En résumé, il ne semble pas y avoir de relation évidente ou triviale entre nœud-connectivité et arc-connectivité.

Modèle linéaire pour la k-connectivité Le modèle linéaire pour la k-connectivité se construit à partir du théorème du flot maximum.

Théorème du flot maximum ou de la coupe minimum On considère d'abord le cas d'un graphe orienté G(V, E) ou V est l'ensemble des nœuds et E l'ensemble des liens. Et on attribue une capacité aux arcs de valeur $c(u,v) = c$, ou c est un réel ou un entier.

On appelle flot une fonction f qui à chaque arc (u,v) du graphe associe une valeur numérique réelle ou entière. Un flot est donc un vecteur de valeurs associé aux arcs ordonnés du graphe. De plus on a les deux contraintes suivantes, toute valeur du flot associée à un arc est inférieure ou égale à la capacité du lien, et d'autre part on a l'application des lois de Kirchoff. On a donc d'une part: $f(u,v) \leq 1,\ \forall (u,v) \in \mathrm{V}$ et d'autre part, tout flot vérifie les lois de Kirchoff c'est à dire que pour tous nœuds v de V, on a égalité des flux entrants et des flux sortants comme l'exprime la relation A.2.

$$\forall v \in \mathrm{V} \sum_{\forall (u,v) \in \mathrm{E}} f(u,v) = \sum_{\forall (v,u) \in \mathrm{E}} f(v,u) \tag{A.2}$$

Aussi bien les algorithmes de calcul de la k-connectivité que les modèles linaires qui sont implémentés sur des solveurs utilisent le théorème de la coupe minimum nommé aussi théorème du flot maximum.

Pour une formulation rigoureuse du théorème on définit un arc u_r fictif particulier dit de retour qui relie les deux sommets source s et destination t doté d'une capacité infinie. Le flot est étendu à cet arc supplémentaire. Et on appelle valeur du flot la valeur du flot sur cet arc supplémentaire: $f(u_r)$. Le théorème du flot maximum exprime l'idée qu'on cherche à maximiser $f(u_r)$. Le théorème peut s'exprimer de la manière suivante:

Théorème 6 *La valeur maximale de $f(u_r)$ pour un flot réalisable dans le graphe sur* $\mathrm{G}(u,v)$ *est égale à la capacité d'une coupe de capacité miminum séparant (déconnectant) s de t. D'autre part, $f(u_r)$ est infinie si il n'existe pas de coupe.*

L'adaptation du théorème à la k-arc-connectivité peut se formuler de la manière suivante, compte tenu des développements précédents.

Corollaire 3 *La k-arc-connectivité entre deux nœuds s et t est égale à la valeur maximale du flot $f(u_r)$, en attribuant une capacité de* 1 *aux arcs du graphe.*

Modèle linéaire Rechercher la coupe minimale revient donc à maximiser la valeur du flot $f(u_r)$ ce qui peut s'exprimer comme un programme linéaire.

Pour cela on a cependant besoin de la matrice d'incidence du graphe qui permet d'exprimer les lois de Kirchoff et la conservation des flots.

La matrice d'incidence du graphe G est définie comme la matrice $n \times m$ de n lignes correspondants au n nœuds du graphe et de m colonnes correspondant aux m arcs du graphe. En notant les nœuds du graphe n avec des indices numérotés n_i, et les arcs du graphe e avec des indices numérotés e_j, $\mathrm{A}_i^j = 1$ si le nœud i est l'extrémité initiale de l'arc u_j, on a pour la matrice d'incidence $\mathrm{A}_i^j = -1$ si le nœud i est l'extrémité finale de l'arc u_j, et sinon $\mathrm{A}_i^j = 0$.

La relation $\mathrm{A} \cdot f = 0$ permet d'exprimer la contrainte visant à respecter les lois de Kirchoff pour les flots.

Finalement le modèle linéaire s'écrit de manière compacte comme en A.3.

$$\begin{aligned} \underset{u_r}{\text{maximize}} \quad & f(u_r) \\ \text{subject to} \quad & \mathrm{A}.f = 0 \end{aligned} \tag{A.3}$$

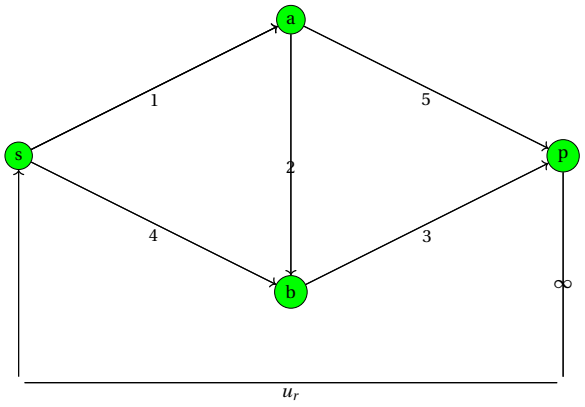


Figure A.2 – Arc retour u_r entre les nœuds s et p

Si on veut appliquer ce modèle basé sur le théorème de la coupe minimum pour trouver la k-arc-connectivité entre deux nœuds s et t d'un graphe orienté, il suffit de poser que la capacité d'un arc vaut 1 et le modèle donnera la k-arc-connectivité entre s et t. Un tel modèle est facilement codable dans un solveur linéaire en nombre entiers.

Algorithmes de calcul de la k-connectivité d'un graphe

k-arc-connectivité d'un graphe Le calcul de la K-arc-connectivité d'un graphe non-orienté G(V, E) consiste généralement à faire appel à un algorithme de recherche du flot maximum pour trouver la coupe minimum du graphe. Comme la coupe minimum du graphe est le minimum de k-arc-connectivité sur toutes les paires de nœuds du graphe, une manière de faire est de calculer effectivement cette grandeur pour toutes les paires de nœuds du graphe, de comparer puis de choisir le minimum. L'algorithme de flot maximum le plus connu est l'algorithme de Ford & Fulkerson.

Dans le cas où la capacité des arcs vaut 1, pour calculer la k-arc-connectivité,

l'algorithme a une complexité bornée en $O(m \cdot f)$, ou f est la valeur de flot maximale. Comme dans un graphe complet la valeur maximale possible d'un flot est $n-1$ (n arcs quittent le nœud source), une borne supérieure est par conséquent $O(m \cdot n)$. Une variation de l'algorithme de Ford & Fulkerson est l'algorithme d'Edmond-Karp qui offre des garanties de convergence que n'offre pas l'algorithme original dans le cas de capacité différentes de 1 ou non entière. Cet algorithme est borné en complexité en $O(n \cdot m^2)$.

On voit que dans ces conditions sur la base de la complexité de l'algorithme de Ford & Fulkerson un algorithme qui testerai systématiquement toutes les paires de nœuds pour trouver la k-arc-connectivité du graphe aurait une complexité bornée en $O(m \cdot n^3)$.

Ce n'est pas très efficace et des améliorations diverses ont été trouvées pour réduire la complexité de calcul.

Titre : Architecture du plan de contrôle SDN et placement de services réseaux dans les infrastructures des opérateurs

Mots clés : SDN, NFV, services réseaux, contrôleurs, placement

Résumé: Le contexte de l'évolution des infrastructures des opérateurs de télécommunications vers les paradigmes SDN et NFV nécessite de lever différents verrous techniques, liés d'une part à la centralisation des fonctions de contrôle, d'autre part aux contraintes d'approches qui s'inspirent directement du Cloud Computing. Dans cette thèse, nous avons abordés deux problématiques. Dans la première nous cherchons à définir une architecture SDN plus adaptée et performante par rapport aux besoins des opérateurs. Pour cela, nous avons proposé un plan de contrôle SDN distribué et flexible visant à dépasser les limites du protocole OpenFlow centralisé ainsi que les contraintes de la virtualisation des fonctions réseaux. L'architecture proposée permet la composition, puis la validation et le déploiement différenciés de services réseaux composables et reconfigurables dynamiquement en prenant en compte les SLA associés aux services. Nous avons illustré certaines propriétés de cette architecture, distribution, composition, dynamicité dans une preuve de concepts. Dans la deuxième, pour réaliser les SLA attendus, nous cherchons à optimiser le placement des services réseaux dans cette infrastructure. Nous avons d'abord traité la problématique du placement de contrôleurs SDN en optimisant des métriques de latence, de charge et de fiabilité, puis de manière plus générale le placement de chaînes de fonctions réseaux virtualisées. Nous avons démontré pour cela les potentialités et les performances des algorithmes évolutionnaires pour tenter de proposer un outil de résolution générique de placement de fonctions réseaux.

Title : SDN control plane architecture and network services placement in Telco infrastructures

Keywords : SDN, NVF, controllers, networks services, placement

Abstract: The evolution of telecommunications operators' infrastructures towards the SDN and NFV paradigms requires to surmount various technical barriers. On one hand, it is necessary to deal with the centralization of control functions, and on the other hand with the constraints of approaches coming directly from Cloud Computing. In this thesis, we addressed two issues. Firstly, we tried to define a SDN architecture more suited to the requirement of operators. For this purpose, we proposed a distributed and flexible SDN control plane to overcome the limitations of the centralized OpenFlow protocol, as well as the constraints of network function virtualization. The proposed architecture allows for the differentiated composition, validation and deployment of dynamically reconfigurable network services, taking into account the SLAs associated with the services. We have illustrated some of its characteristics, namely, distribution, composition, dynamicity in a proof of concept. Secondly, to achieve the expected SLAs, we try to optimize the placement of network services in this infrastructure. We first dealt with the issue of SDN controllers placement seeking for the optimization of latency, load and reliability metrics. Then, we considered the placement of virtualized network functions chains. We have therefore demonstrated the potentialities and performances of evolutionary algorithms with the perspective to propose a generic resolution tool for placement of network functions.